Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Estudio de la plataforma domótica Home Assistant e integración en Raspberry Pi

Autor: Santiago Romero Cabrera
Tutor: José María Maestre Torreblanca
Cotutor: Jesús Iván Maza Alcañiz

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías de Telecomunicación

Estudio de la plataforma domótica Home Assistant e integración en Raspberry Pi

Autor:
Santiago Romero Cabrera

Tutor:
José María Mestre Torreblanca
Profesor titular
Cotutor:
Jesús Iván Maza Alcañiz

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Estudio de la plataforma domótica Home Assistant e integración en Raspberry Pi

Autor: Santiago Romero Cabrera

Tutor: José María Maestre Torreblanca

Cotutor: Jesús Iván Maza Alcañiz

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis amigos

A Marisa y Emilio

Agradecimientos

Mis más sinceros agradecimientos son a mi familiares y amigos, personas que desde que inicié mi andadura en una ingeniería han estado apoyándome para pasar los momentos más difíciles y brindando en los momentos más felices. A José Manuel, amigo y compañero de grado con el que tantas tardes y clases he compartido. A Marisa y Emilio, pilares fundamentales en el desarrollo de este proyecto por su paciencia y comprensión. Por último y no menos imporante a mis tutores, José María e Iván Maza, por su flexibilidad y adaptación para poderme permitir hacer el trabajo a distancia.

Santiago Romero Cabrera

Sevilla, 2019

Resumen

El objetivo de este proyecto es el estudio del software Home Assistant, su integración en una Raspberry Pi, y mediante la compra de distintos dispositivos, conformar un sistema domótico en casa mediante la integración de los mismos. El documento está divido en varias partes en las que se sigue un hilo para poder entender de manera secuencial que se quiere hacer.

La primera parte nos habla sobre el paradigma de la domótica, haciendo un repaso a sus fundamentos y dando una visión sobre las posibilidades a mano del usuario. En la segunda parte del documento se presentan, las que a mi parecer son las plataformas domóticas de código abierto más importantes. En el tercer punto se hace una breve introducción a la computadora utilizada en el proyecto, una Raspberry Pi. Los siguientes puntos que son el número cuatro y cinco se centran en el núcleo principal del trabajo que es la explicación del funcionamiento interno de HA, y su utilización a modo de usuario para poder integrar dispositivos y jugar con ellos. En el último punto se detallá el sistema domótico que he empezado a hacer en mi hogar, en el que se muestra paso a paso lo necesario para integrar los dispositivos que voy nombrando.

Abstract

The objective of this project is the study of the Home Assistant software, its integration into a Raspberry Pi, and through the purchase of different devices, forming a home automation system by integrating them. The document is divided into several parts in which a thread is followed in order to understand sequentially what you want to do.

The first part tells us about the home automation paradigm, reviewing its fundamentals and giving a vision about the possibilities at the user's hand. In the second part of the document are presented, which in my opinion are the most important open source home automation platforms. In the third point a brief introduction is made to the computer used in the project, a Raspberry Pi. The following points, which are number four and five, focus on the main core of the work, which is the explanation of the internal functioning of HA, and its use as a user to be able to integrate devices and play with them. In the last point, the home automation system that I have begun to do in my home is detailed, which shows step by step what is necessary to integrate the devices I am naming.

... -translation by google-

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

HA	Home Assistant
IU	Interfaz de usuario

1 INTRODUCCIÓN

Vivimos en un mundo hiperconectado y la casa no es ajena a ello.

- Equipo redacción de "Houzz" -

En este proyecto se abordará el tema de la domótica y en concreto una plataforma de código abierto que hará de nuestro hogar un lugar más inteligente a un precio moderado. Pero, antes de esta plataforma, al igual que otras de código abierto, ¿dónde se enmarcaba la domótica?, ¿Qué es un hogar digital? ¿Qué es propiamente la domótica? ¿Cuál es su objeto de estudio? En este apartado introductorio se intentará responder estas preguntas haciendo una introducción al susodicho término.

1.1 Objetivos

Los **objetivos** de este proyecto son claros, lo primero es asentar las bases un tema que en mi opinión es fascinante como es la domótica haciendo una introducción a los aspectos y protocolos más significativos. Tras esto, y antes de hablar del núcleo del trabajo, es necesario dar una visión de cómo está el mercado con respecto a las plataformas domóticas de código abierto, para dar a entender que no solo existe una única solución posible, y que como en cualquier tecnología tenemos varias posibilidades al alcance con sus pros y contras. En el núcleo de mi trabajo presento una plataforma que he descubierto para hacer este proyecto (Home Assistant), a raíz de una que se presentó en la asignatura de "Domótica" del 4º curso de GITT (OpenHab). En este documento hago una presentación de la plataforma a dos niveles, una primera parte sobre la arquitectura de la misma, y otra sobre como poder utilizarla a nivel de usuario, destacando los aspectos más importantes. Lo último que se expone en este documento es una prueba práctica real, es decir, crear desde cero un sistema domótico "Mi sistema domótico" poniendo a prueba lo aprendido.

Una vez presentados los objetivos, y para entrar en una previa a Home Asisstant dejo una introducción.

1.2 Reseña histórica

La domótica se inicia a comienzos de los años setenta cuando aparecieron los primeros dispositivos de automatización en edificios a base de pruebas pilotos. Pero fue en la década de los 80 cuando los sistemas integrados se utilizaron a nivel comercial, para luego desarrollarse en el aspecto doméstico de las casas urbanas.

En 1975 aparece la tecnología X10, protocolo de comunicaciones para el control remoto de dispositivos eléctricos que utilizan la línea eléctrica preexistente. Este sistema se extendió por Estados unidos y Europa. Avanzando en el tiempo en 1997 y mediante asociaciones, los tres sistemas de automatización europeos más

arraigados (Batibus, EHS y EIB) comenzaron un proceso de convergencia hacia la asociación Konnex, que finalmente acabo en la asociación KNX, un estándar de protocolo de comunicaciones de red, basado en OSI, para edificios inteligentes. También por los 90 aparece la tecnología LonWorks, plataforma tecnológica basada en el protocolo abierto llamado LonTalk, para aplicaciones de control y automatización. A partir del año 2000 hubo un fuerte incremento en las empresas como fabricantes especializados en productos del estándar KNX y LON, cubriendo nuevos productos, compartiendo precios y haciendo más fuerte al protocolo estándar que eligen.

En el año 2006 empiezan a surgir los sistemas domóticos inalámbricos, usando como protocolos Zigbee y Zwave abriendo al abanico de la implantación de los mismos para viviendas ya que las funciones de los sistemas cableados no son capaces de cometer por si solos como por ejemplo la instalación sencilla.

Hoy en día, la domótica aporta soluciones dirigidas a todo tipo de viviendas y se ofrecen más funcionalidades por menos dinero, más variedad de producto, y gracias a la evolución tecnológica, son más fáciles de usar y de instalar. Todo esto es posible en parte a la cantidad de fabricantes que ofrecen productos domóticos, que son compatibles con plataformas de control gratuitas como Home Assistant.

1.3 Introducción a la domótica

Podríamos definir la domótica como el conjunto de servicios proporcionados por sistemas tecnológicos para satisfacer las necesidades básicas de seguridad, comunicación, gestión energética y confort, del hombre y de su entorno más cercano. Bajo el concepto de domótica nos vamos a referir a los mecanismos de automatización y control (apagar / encender, abrir / cerrar y regular) de los sistemas domésticos como la iluminación, climatización, persianas y toldos, puertas y ventanas, cerraduras, riego, electrodomésticos, suministro de agua, suministro de gas, suministro de electricidad, etc.

El término domótica nace del neogolismo francés ‘domotique’, el cual procede de la palabra latina domus (casa) y del francés telematique (telecomunicación-informática).

Las posibilidades que nos brinda la domótica son la siguientes

- Mayor confort
- Aumento de la seguridad
- Control del gasto energético
- Mayor capacidad de ocio

Figura 1-1. Posibilidades domóticas.

La domótica consiste en la integración de componentes, perdiendo así la visión de soluciones aisladas no que no cooperan entre sí. Actualmente en el mercado existen dos tipos de soluciones.

- Gran cantidad de pequeños aparatos, que tienen un coste barato y con dedicación específica. No están sujetos a estándar
- Grandes sistemas domóticos diseñados para un campo de aplicación definido. Su diseño está guiado para satisfacer una serie de necesidades, pero tienen un alto coste.

Mi proyecto personal dómotico, tal y como se podrá observar en el punto 6, consta de lo descrito en el primer punto, "gran cantidad" de pequeños aparatos, con un coste no muy elevado con capacidad de integración en un mismo entorno.

1.3.1 Funciones de los sistemas domóticos

Gestión de la energía: mejora el uso de un recurso escaso como es la energía. Ejemplo de ello puede ser el control de la iluminación interior y exterior, gestionar el consumo de agua para no saltar al siguiente bloque de tarificación …

Automatización de tareas domóticas: subir/bajar persianas en función de la cantidad de luz recibida, activación de la climatización según la temperatura. En mi sistema, se describen las diferentes automatizaciones realizadas para la gestión de luces, y otros componentes.

Seguridad: protección contra robos, detección de incendios, aviso a organismos de seguridad. Es uno de los grandes objetivos de la domótica, y como no podía ser menos, aunque a muy bajo nivel y con los recursos que tengo, en mi sistema con un sensor de puerta y ventana propongo una solución de seguridad.

Monitorización de la salud: llevar a cabo una vigilancia de la salud de una persona donde el sistema sea capaz de identificar una situación de riego …

Control remoto de la vivienda: apagar/encender distintos dispositivos de una vivienda. De nuevo, y en referencia a mi proyecto, se tiene el control remoto de varias luces en la vivienda. Lo mejor que ofrece HA es que pueden ser de diferentes fabricantes.

Control remoto desde fuera de la vivienda: se lleva a cabo a través de un móvil o cualquier aparato con conexión a internet. Este es un punto desarrollando en 6.4 en el que se podrá acceder a la aplicación de "*santi home*" desde incluso fuera de la red local.

1.3.2 Red de interior y pasarela residencial

Los dispositivos se conectan a través de una red interna llamada "Home Area Network" HAN. Es un tipo de red de área local (LAN) que se desarrolla a partir de la necesidad de facilitar la comunicación y la interoperabilidad entre los dispositivos digitales presentes en el interior o en las inmediaciones de una casa.

Los dispositivos capaces de participar en esta red a menudo logran mayores capacidades emergentes a través de su capacidad para interactuar. Estas capacidades adicionales se pueden utilizar entonces para aumentar la calidad de vida dentro de la casa en una variedad de formas, tales como la automatización de las tareas repetitivas, aumento de la productividad personal, la seguridad doméstica mejorada y un acceso más fácil al entretenimiento.

La HAN se puede dividir en tres tipos de redes:

- Red de control: interconecta los sensores, actuadores y electrodomésticos inteligentes con el sistema de control.
- Red de datos: interconecta el PC, impresoras, escáneres, etc.
- Red multimedia: permite la distribución de audio y video por toda la casa. Interconecta los dispositivos multimedia como pueden ser los televisores, radios, cámaras de vídeo…

La HAN debe conectarse al exterior mediante algún tipo de tecnología habilitadora para ello como puede ser la fibra óptica o el ya casi en desuso ADSL.

La pasarela residencial es un dispositivo que permite la convivencia de todas las redes y dispositivos internos entre sí y el exterior y que garantiza la seguridad de las comunicaciones hacia/desde la HAN. Debe poder ser gestionable de forma remota.

1.3.3 Elementos domóticos

Podemos dividir los elementos básicos que conforman un sistema domótico en sensores, sistema de control y actuadores.

1. Sensores

Son elementos que recogen la información del entorno como puede ser la temperatura, humedad, cantidad de luz, … y la envían al sistema de control centralizado para que actúe en consecuencia. En algunos casos los sensores pueden comunicarse directamente con los actuadores. A la hora de la instalación suelen ser flexibles ya que no se conectan a la corriente eléctrica y funcionan por baterías.

Una lista de los más comunes:

- Termostato ambiente
- Detector gas
- Detector incendios
- Sensor humedad
- Sensor de iluminación
- Sensor de presencia
- Sensor de temperatura

En mi sistema domótico he usado, un sensor de movimiento, otro de temperatura y por último uno de humedad.

2. Actuadores

Son dispositivos que utiliza el sistema de control para modificar el estado de equipos o instalaciones. Un ejemplo de ello es un sensor de humo que detecta un incendio, avisará al sistema de control y este hará las llamadas telefónicas programadas y actuará sobre la válvula de corte de gas de la vivienda y sobre la sirena.

Los actuadores más comunes:

- Contactores
- Electroválvulas
- Sirenas o elementos zumbadores

Como actuador presentado en mmi aplicación práctica suelen ser las bombillas, que son las que reciben órdenes de automatización.

3. Sistema de control

La arquitectura de control del sistema se puede llevar a cabo de tres formas diferentes:

- Centralizada: este sistema está organizado de tal forma que el controlador sea el eje central del sistema, recibiendo la información de los sensores, analizándola, y enviado órdenes a los actuadores, según la configuración, o la información que reciba por parte del usuario. Los elementos a controlar y supervisar han de conectarse al sistema de control y si dicho sistema falla, todo deja de funcionar.

  Esta es la arquitectura que sigue mi proyecto, todos los elementos que integro, como bombillas, sensor de presencia, enchufe pasan por un controlador que recoge la información y según las reglas detalladas en él, actúa en consecuencia. Mi centro de control será una Raspberry Pi, donde se estará ejecutando Home Assitant.

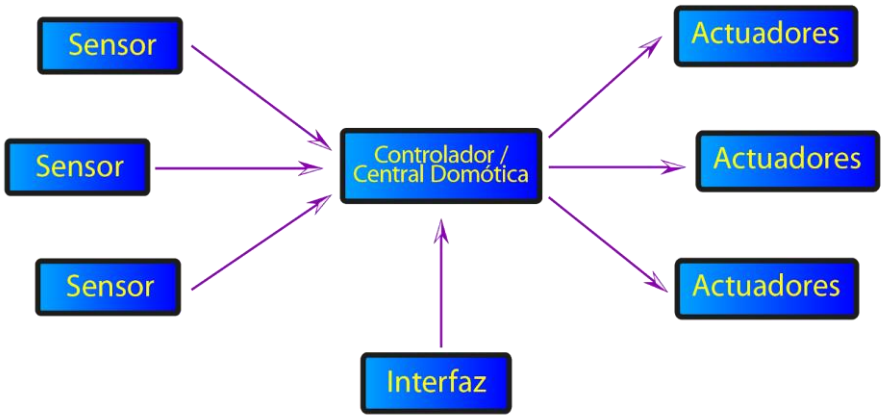


Figura 1-2. Arquitectura centralizada.

- Distribuida: este tipo de arquitectura se diferencia por tener sensores y actuadores que son a su vez controladores, es decir, son capaces de analizar información, y están conectados a través del bus central.

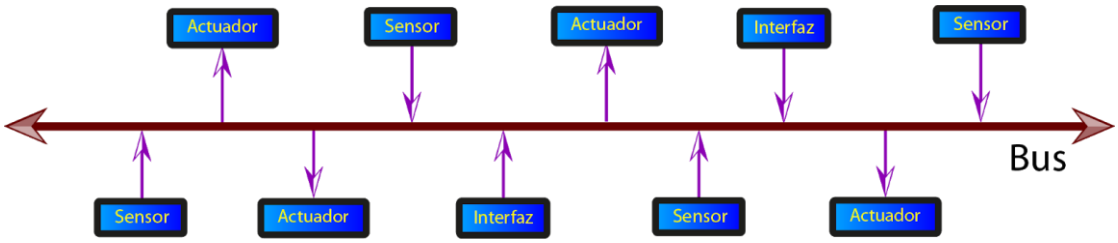


Figura 1-3. Arquitectura distribuida

- Mixta: se distribuye la instalación a controlar en zonas y cada una de controla de forma de centralizada

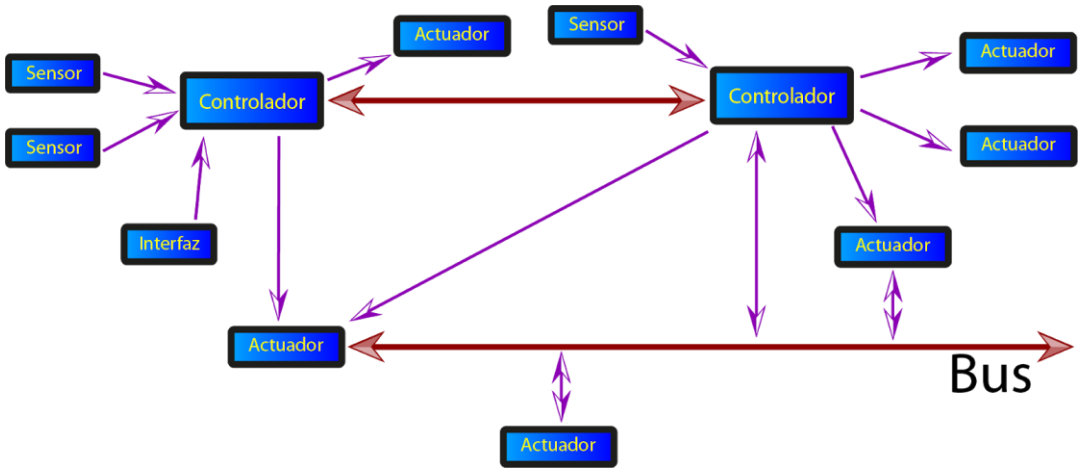


Figura 1-4. Arquitectura mixta.

Los usuarios pueden interactuar con el sistema de distintas maneras, es decir, existen distintas interfaces de usuario.

- Interfaz local: la centralita incorpora una pantalla y un teclado
- Interfaz de voz: permite programar o conocer el estado desde cualquier dispositivo que implemente sistema de control audio-voz. Aunque mi proyecto no lo incorpora sería posible darle la posibilidad de una interfaz por voz mediante *Google Home*, u otros asistentes como *Alexa*
- Interfaz web: El sistema dispone de un servidor web que permite configurar o conocer el estado actual de una forma gráfica (HTTP). Es la manera en la que accedo a la interfaz gráfica de HA.

1.4 Protocolos

La tecnología en redes domóticas se basan en el uso de protocolos de comunicación y de estándares para la comunicación entre dispositivos. A continuación, se expondrán los protocolos de comunicación utilizados en domótica que han tenido más fuerza en el mercado.

A parte de lo que es el protocolo, cada uno tiene sus propios productos, sensores, etc… y la gran potencia que ofrece HA es que es capaz de integrar en una misma casa productos de KNK, Z-Wave, y X-10. Los productos de LonWorks no se recogen dentro de HA.

1.4.1 KNK

KNX es un estándar de protocolo de comunicaciones de red, basado en OSI, para domótica. Se crea gracias a la cooperación y convergencia de tres estándares: el European Home Systems Protocol (EHS), el European Installation Bus (EIB o Instabus) y el BatiBUS pertenecientes, respectivamente, a la EHSA (European Home Systems Association), la EIBA (European Installation Bus Association) y el BCI (BatiBUS Club International). El estándar KNX está gestionado por la Asociación KNX. Es un estándar abierto desde 2016.

KNX incluye 4 medios distintos de trasmisión:

- Par trenzado
- Ondas portadoras
- Radio frecuencia
- IP

1.4.2 LonWorks

LonWorks es una plataforma de control que permite diseñar sistemas de control distribuidos, escalables y fácilmente compatibles con otros sistemas. LonWorks está pensado para la automatización a gran escala.

Se trata de una tecnología de red abierta y ofrece soluciones extremo a extremo basadas en una arquitectura descentralizada.

LonWorks utiliza para el intercambio de información (ya sea de control o de estado) el protocolo LonTalk. Este tiene que ser soportado por todos los nodos de la red. Toda la información del protocolo está disponible para cualquier fabricante. También existe un organismo que evalúa la compatibilidad de los productos del estándar llamado LonMark.

El protocolo LonWorks es independiente del medio, permitiendo a los equipos LonWorks comunicarse sobre cualquier medio de transporte físico.

1.4.3 X-10

X10 es un protocolo de comunicaciones para el control remoto de dispositivos eléctricos que utiliza la línea eléctrica para transmitir señales de control entre equipos de automatización del hogar. Los dispositivos X10 que se comercializan son solo para uso individual y en entornos domésticos de hasta 250 m², dada su limitación en ancho de banda y en el número máximo de dispositivos a controlar (256). No obstante existen elementos de última generación que incorporan, entre otros, los protocolos X-10 extendidos, para dar funcionalidad a soluciones de comunicación como la capacidad de bidirección, solicitud de estados y comprobación de la correcta transmisión de las tramas.

1.4.4 Z-Wave

Es una tecnología inalámbrica para la domótica que permite que todos sus aparatos electrónicos en el hogar o en el trabajo hablen unos con otros. Usa señales de radio de baja potencia que viajan fácilmente a través de paredes, pisos y muebles sin ser interferidas por dispositivos inalámbricos que pueda tener en funcionamiento. Se inició como un protocolo propietario, pero en 2012 se convirtió es un estándar abierto.

Z-Wave unifica todos sus aparatos electrónicos y los integra en una sola red, sin complicadas programaciones y sin instalar complicados cableados. Cualquier dispositivo habilitado con Z-Wave puede ser añadido fácilmente a una red, y otros dispositivos que no están habilitados con Z-Wave pueden volverse compatibles, tan solo conectando un módulo.

Existen otros protocolos inalámbricos que compiten con Z-Wave, como son el Wi-Fi pero la ventaja de Z-Wave es que trabaja a una frecuencia de 900 MHz, en lugar de a 2,4 GHz. El hecho de trabajar a 900MHz proporciona un rendimiento superior por dos motivos: menos interferencias (por funcionar a baja frecuencia) y mayor penetración de las ondas en paredes, pisos y muebles (al tener mayor longitud de onda).

Cada red Z-Wave puede incluir hasta 232 dispositivos Cada nodo alimentado, por lo general, es capaz de retrasmitir un mensaje recibido para así garantizar la conectividad en lo que se conoce como una red mallada inteligente. El rango de comunicación medio entre dos nodos alcanza los 30 metros (100 metros máximo), con la capacidad de repetir el mensaje hasta 4 veces o saltos entre nodos.

2 PLATAFORMAS DOMÓTICAS

Hace unos años los sistemas domóticos para el hogar estaban al alcance de muy pocas familias debido a al alto coste económico que ofrecían las empresas por los sistemas que te proporcionaban. Con el paso del tiempo, el alza de la tecnología y la multitud de dispositivos conectados (IoT) con un precio más asequible, ha sido posible crear viviendas conectadas capaces de ofrecernos multitud de opciones.

Aunque las tecnologías se encuentran en un estado bastante avanzado, existe un problema y una limitación a la hora de diseñar sistemas domóticos debido a la ausencia de interfaces enfocadas a usuarios con diferentes necesidades y a la baja interoperabilidad entre los fabricantes de los distintos sistemas de automatización de viviendas y edificios. Esta situación nos hace centrarnos en un único fabricante, en lugar de adaptar bajo una misma interfaz y sistema de control diferentes soluciones domóticas según sean nuestras necesidades. Por ejemplo, podríamos necesitar combinar las tecnologías domóticas de un fabricante y las tecnologías de otro.

Debido a estas necesidades nacieron las plataforma domóticas de código abierto que se pueden descargar y cargar en ordenadores de bajo costo como puede ser una Raspberry Pi. Estás plataformas están diseñadas para integrar diferentes sistemas de automatización de viviendas, dispositivos y tecnologías dentro de una misma solución.

De las varias existentes en el mercado, en este documento se exponen tres que a mi parecer son las más interesantes y extendidas.

2.1. Domoticz

Domoticz es un software libre de control domótico disponible para las plataformas Windows y Linux y caracterizado por consumir muy pocos recursos del sistema lo que lo convierten en una solución muy interesante si queremos combinarlo con un algún controlador de bajo coste como puede ser una Raspberry Pi.

Domoticz ofrece soporte a un gran número de protocolos domóticos como pueden ser EnOcean, X10 o Z-Wave así como una buena integración con diferentes dispositivos inalámbricos como mini estaciones meteorológicas o cámaras IP.

Cuenta además con una interfaz web y aplicaciones móviles que convierten su consumo multiplataforma en una tarea sencilla.

Es uno de los primeros softwares de este tipo por lo que tiene un amplio recorrido y experiencia en este campo. Como desventaja la interfaz está algo desactualizada respecto las tendencias actuales y, lo que es más importante, los diferentes módulos que lo componen están poco desacoplados del "core" del sistema, lo que tiene la principal repercusión de un complicado desarrollo de nuevos módulos y funcionalidades.

Esta situación ha llevado a que el número de módulos y tecnologías disponibles, aunque amplio, sea algo menor que otras opciones.

El software está escrito en C++ y la documentación no resulta tan completa como sus competidoras.

Figura 2-1. Interfaz y logo de Domoticz.

2.2. OpenHab

OpenHAB es una de las plataformas Open Source que cubre una gran cantidad de sistemas: Windows, OSX, Linux, sistemas embebidos como NAS, etc.

Su diseño se basa en una arquitectura totalmente modular donde tenemos el core del sistema y sobre el que se construyen alrededor los diferentes módulos que le añaden funcionalidad. Está escrito en Java y su modularidad ha permitido que cuente con muchos módulos diferentes que cubren gran cantidad de elementos hardware y protocolos.

Con OpenHAB podríamos trabajar con relativa sencillez con KNX, ZWave, X10, el termostato Nest, bombillas HUE, bases de datos, servicios Web como IFTTT o una API que nos sirva el tiempo y la previsión. Tiene un buen diseño visual y consta de bastante documentación, tanto del Core como de los módulos accesorios. Cuenta con un diseño que adapta la disposición de los elementos de los paneles de control al tamaño de la pantalla lo que facilitan su consumo y visualización multiplataforma. Además, cuenta con aplicaciones oficiales para iOS y Android.

Cuenta con una amplia comunidad que, una vez más se reúnen alrededor del foro oficial y que en muchos casos es la referencia oficial para solucionar los errores con los que nos encontremos y que resaltan las carencias de la documentación en algunos apartados.

Precisamente para su configuración cuenta con un entorno preparado que deriva de Eclipse (totalmente multiplataforma) y que busca facilitar la vida al instalador /configurador del sistema.

Figura 2-2. Interfaz y logo de OpenHab.

2.3. Home Assistant

Home Assistant es una plataforma open-source de automatización del hogar creada en Python 3 que permite, a través de diferentes módulos, la interacción con diferentes plataformas, servicios y dispositivos. Es el que ocupa mi proyecto y vamos a estudiarlo más en profundidad a lo largo del proyecto.

Cuenta con 1442 componentes (con fecha de junio 2010) disponibles perfectamente organizados en varias categorías como "Climate", "DIY", "Health", "Social", "Notifications" o "Weather". La idea de los componentes es proveer una interfaz de comunicación entre la plataforma y diferentes elementos externos que pueden ser plataformas online como IFTTT, iCloud o APIs de terceros, sistemas que comparten un elemento físico y una plataforma cloud como pueden ser Nest o las Philips HUE, elementos puramente físicos que conectamos directamente o a través de un protocolo como MQTT y elementos software como por ejemplo una plataforma que nos permite emitir notificaciones a bots de Telegram.

Figura 2-3. Interfaz y logo de HA.

3 RASPBERRY PI

En nuestro trabajo se usará una Raspberry Pi como controlador de nuestra casa. Pero ¿qué es exactamente?

Raspberry Pi es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste. En cuanto a su precio, suele estar por debajo de los 40 euros, una de las razones que explica su popularidad.

Para que funcione, basta con que añadamos nosotros mismos un medio de almacenamiento (como por ejemplo una tarjeta de memoria SD), enchufarlo a la corriente gracias a cualquier cargador de tipo microUSB (el mismo que sirve para recargar la mayoría de los teléfonos móviles, cuyo coste es ínfimo) y, si lo deseamos, incorporar un chasis para que todo quede a buen recaudo y su apariencia sea más estética. Estos pueden ser desde cajas predeterminadas hasta una que fabriquemos nosotros mismos echándole grandes dosis de imaginación.

La fundación de Raspberry Pi pone a disposición desde su página web Raspbian, una distribución de Linux basada en Debian, pero también podemos recurrir a muchas de las distribuciones específicas que la comunidad de usuarios ha desarrollado para diversos fines. La que usaré recibe el nombre de Hassbian.

En función del modelo que escojamos, dispondremos de más o menos opciones de conexión, pero siempre tendremos al menos un puerto de salida de video HDMI y otro de tipo RCA, minijack de audio y un puerto USB 2.0al que conectar un teclado y ratón.

En cuanto a conexión de red se refiere, podemos disponer de Ethernet para enchufar un cable RJ-45 directamente al router o recurrir a adaptadores inalámbricos Wi-Fi, pero en las versiones posteriores a la Raspberry Pi 2 ya encontramos una tarjeta Wi-Fi preinstalada.

Los modelos existentes son los siguientes:

Raspberry Pi 1 modelo A

Este fue el primer modelo de Raspberry, sus ventas comenzaron en el año 2012. Carecía de puerto Ethernet, por lo que para su conexión a Internet requería de un adaptador Wi-Fi por USB. Poseía 26 conectores GPIO, salida de vídeo vía HDMI y Video RCA, un conector Jack de 3.5 milímetros, un único conector USB, MicroUSB (De alimentación) y un conector de cámara. Su procesador fue un Broadcom BCM2835, Single-Core a 700MHz. También tuvo 256 MB de RAM y una gráfica Broadcom VideoCore IV. Requería de una fuente de alimentación de 5 voltios y 2 amperios, elemento común al resto de versiones. Tuvo un coste inicial de 40 euros.

Raspberry Pi 1 modelo B y B+

También del año 2012, es una variante del Modelo A, trajo consigo diversas mejoras, la inclusión del doble de memoria RAM, pasando de 256MB a 512MB. Trajo consigo un puerto USB más y, por fin, un conector Ethernet (RJ-45) Se mantuvo tanto su tamaño como su coste. No hubo variaciones ni en el procesador ni en la parte gráfica. Tiempo después se lanzó el Modelo B+, que incluyó 4 puertos USB y pasó de usar una SD a una MicroSD.

Raspberry Pi 2 modelo B

Lanzada en 2014 es el primer modelo que no incluye el mismo procesador usado en los tres anteriores: se sustituye por uno de la misma marca, pero de modelo BCM2836. Pasa de ser de un núcleo a cuatro, y de 700MHz a 900MHz. No obstante, emplea la misma gráfica, la VideoCore IV. Dobla la cantidad de memoria RAM, pasando de 512MB a 1GB (Algo menos en realidad) esta memoria está compartida con la gráfica. También incluye 40 pines GPIO, y mantiene los cuatro puertos USB. Suprime la conexión RCA.

Raspberry Pi 3 modelo B

Sacada a la luz en el año 2016, renueva procesador, una vez más de la compañía Broadcom, una vez más un Quad-Core, pero pasa de 900MHz a 1.20GHz. Mantiene la RAM en 1GB. Su mayor novedad fue la inclusión de Wi-Fi y Bluetooth (4.1 Low Energy) sin necesidad de adaptadores.

Raspberry Pi 3 modelo B+

La Raspberry Pi 3 B+ apareció en marzo del 2018 para actualizar el modelo anterior la Raspberry Pi 3 Model B y entre sus mejoras cuenta con un nuevo procesador y mejor conectividad,[57] así que pasa de tener 1.2Ghz a tener 1.4Ghz y en cuanto a la conectividad inalámbrica ahora incorpora doble banda a 2,4GHz y 5GHz, y su nuevo puerto Ethernet se triplica, pasa de 100 Mbits/s en el modelo anterior a 300 Mbits/s en el nuevo modelo, también cuenta con Bluetooth 4.2 (Low Energy).

Raspberry Pi 3 modelo A+

Fue anunciada en Noviembre de 2018.[58] Los modelos A+ presentan menores prestaciones a un menor precio. Cuenta con 512 MB de RAM (compartidos con la GPU VideoCore IV), un solo puerto USB y sin puerto de conexión de red por cable (RJ-45).

Raspberry Pi 4 modelo B

Ha sido anunciada este Junio de 2019. Se han cambiado los puertos HDMI de tamaño completo por dos puertos microHDMI. Cuenta con la capacidad de manejar dos pantallas 4K a 60 Hz. Se ha incluido por primera vez USB 3.0, y el puerto Ethernet ya no está limitado a 300 Mbps. Tiene un procesador Broadcom nuevo hasta tres veces más eficiente que el anterior.

En nuestro proyecto hemos utilizado la Raspberry Pi 3 modelo B+

Figura 3-1. Raspberry Pi3 3 modelo B+.

Para el almacenamiento, Raspberry Pi recomienda utilizar una tarjeta SD con una capacidad mínima de 4 Gbytes y de clase 4. He usado en mi proeycto una tarjeta Samsung evo 32GB de clase 10.

Para enchufar nuestra Raspberry Pi a un monitor o televisor, necesitaremos un cable HDMI o, si no disponemos de tal entrada de video, un cable HDMI a DVI. También es posible recurrir en su lugar a la salida analógica RCA

Centrándonos en el teclado y ratón, lo más sencillo es adquirir un conjunto inalámbrico que conectaremos mediante un único adaptador, y así no necesitaremos de más puertos USB.

El sistema operativo más común para utilizar es Raspbian y es una distribución del sistema operativo GNU/Linux basado en Debian, y por lo tanto libre para la SBC Raspberry Pi, orientado a la enseñanza de la informática

4 HOME ASSISTANT

4.1 Preámbulo

Antes de meternos de lleno en el funcionamiento y arquitectura de esta plataforma domótica he querido exponer, el por qué es necesaria y ver el paradigma años atrás cuando todo era un barullo de conceptos.

Desde que la tecnología se hizo tan accesible y diminuta se ha podido acceder a la automatización del hogar como algo más propio, más accesible al consumidor. Muchos términos diferentes años atrás empezaron a toar fuerza como el Internet de las cosas, la automatización del hogar y el hogar inteligente.

El **Internet de las Cosas** (IoT) ha creado una generación de dispositivos que no solo pueden ser controlados por personas a través de botones o controles remotos, sino que también proporcionan una interfaz para comunicarse con otros dispositivos y aplicaciones. Por ejemplo, una bombilla puede recibir comandos para crear diferentes tipos de escenas de iluminación y regular diferentes intensidades.

No existía un estándar abierto ampliamente adoptado para la comunicación de dispositivos inteligentes. Esto evitaba que muchos dispositivos se comunicarán entre sí e incluso si pudieran, la mayoría de los dispositivos no están diseñados para administrar otros dispositivos. Para resolver esto, necesitamos un dispositivo para poder comunicarnos y administrar todos estos dispositivos conectados. Este dispositivo se llama un **hub**.

Como mínimo, un hub debe realizar un seguimiento del estado de cada dispositivo y debe ser capaz de controlarlos si es posible. Por ejemplo, tiene que saber qué luces están encendidas o apagadas y ofrecer una forma de controlarlas. Para un sensor solo hay que conocer el valor. Un centro con estas capacidades ofrece el control de casa.

4.2 Home assistant como hub

No existe un estándar extensamente adoptado para la automatización de las viviendas. Los vendedores, los de automatización del hogar, por lo que he podido comprobar no adoptan el soporte a pequeños estándares cuando van al mercado. La solución propia y local a esto, es adoptar plataformas de código abierto como Home Assistant.

Hoy en dia se pueden comprar componentes como bombillas, termostatos, reproductores multimedia y controlarlos con sus aplicaciones. Las luces no saben acerca del termostato, tampoco saben acerca del reproductor media. Como usuario tienes una APP por cada dispositivo que quieres controlar, para subir la temperatura del termostato, para controlar las luces, para reproducir distintos dispositivos media. Con este modelo, es imposible que los dispositivos que tenemos en casa sepan acerca de los demás dispositivos, es decir, no existe comunicación por lo que es imposible crear reglas de automatización. Para solucionar este problema tenemos una cosa llamada hub, un “home automation hub” y esto es lo que Home Assitant es. Con este hub tenemos todos esos dispositivos hablando entre ellos con un protocolo diferente.

Cuando atendemos a la esencia de lo que es un hub tenemos que pensar en los siguientes conceptos.

Home control: es el bloque de donde saldrán las distintas automatizaciones para que los dispositivos puedan hablar y comunicarse.

IoT: son los distintos dispositivos y sensores que mandan la información

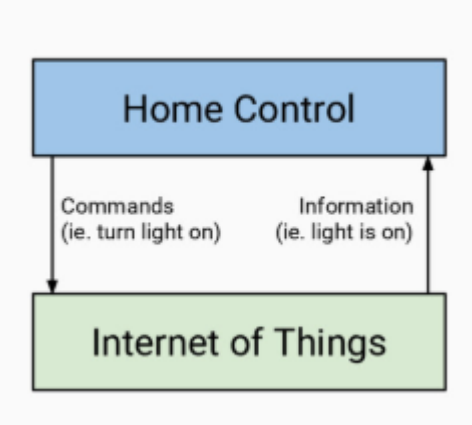


Figura 4-1. Esencia de un hub.

La esencia del proceso es muy simple, lo primero es comunicarse con otras cosas enviando comandos. "Termostato, establece esta temperatura". "Luz, enciéndete". "Reproductor media, reproduce esta película". Lo segundo es obtener información de las cosas (Internet de las Cosas) la cual se puede dividir con dos términos fundamentales. El primer término es el **estado**. Los dispositivos tienen un estado y en cualquier momento del tiempo se podría mirar al dispositivo y consultar su estado. Por ejemplo, en una luz su estado es encendido y pueden se pueden tener atributos extra que expliquen el estado del dispositivo, como vemos en la imagen el estado es encendido y tiene un brillo del 60% y su color es rojo. Si el reproductor está funcionando, el volumen a 80% y tenemos puesto juego de tronos.

La otra parte de información que podemos obtener del internet de las cosas son los **eventos**. Los eventos no es algo que se pueda obtener como un estado, es decir, no hay una propiedad que los defina, para detectar un evento se tiene que fijar la atención en un punto en concreto. Por ejemplo, una luz está siendo encendida o un movimiento ha sido detectado.

- Estados
  - Luz = encendida
    - Brillo = 60%
    - Color = rojo ("on")
  - Reproductor = encendido
    - Volumen= 80%
    - Programa TV = Juego de tronos
- Eventos
  - Estado de la luz ha cambiado
  - Movimiento detectado

Lo que se acaba de explicar es el funcionamiento de un hub, y cómo nosotros como usuarios podemos tener el control de casa.

¿Cuál es el núcleo de la automatización del hogar? Atendamos a esta frase para la explicación "cuando llegue a casa y el sol se ponga quiero que las luces se enciendan". Esto es la automatización del hogar, en este tipo de reglas reside el corazón del asunto.

La automatización del hogar está basada en reglas:

- El disparador, o accionado ("trigger") es siempre un evento. No confundirlo con, por ejemplo, cuando las luces estén encendidas ("on") se dispara tal automatización, porque las luces pueden durar

encendidas un periodo largo de tiempo por lo que se estarían accionando automatizaciones todo ese tiempo. Por lo que lo correcto es estar atentos al momento que se enciendan las luces, es decir, al evento en sí. En la frase dada anteriormente para la explicación de este apartado un evento sería "cuando llegue a casa" es decir, como se verá en mi sistema domótico cuando mi dispositivo móvil pase del estado de *not_home* a *home*.

- Condiciones: el usuario no siempre desea que las reglas de automatización del hogar se ejecuten, es decir, en ciertas ocasiones necesitamos de distintas casuísticas para su ejecución. En el ejemplo dado la condición es "cuando el sol se ponga" es cuando realmente quieres que las luces se pongan.

- Envío de comandos: Enciende las luces.

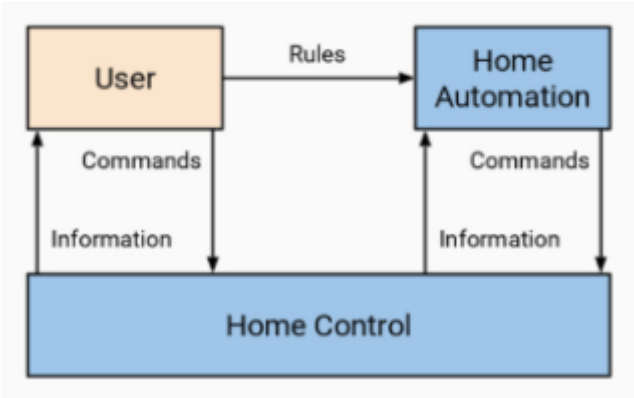


Figura 4-2. Reglas de usuario.

4.3 Arquitectura HA

4.3.1 Introducción

Antes de adentrarnos en la arquitectura de Home Assistant, obtengamos una visión general clara del panorama de la automatización del hogar en su conjunto. De esta manera, podemos mostrar cómo las diferentes partes de Home Assistant encajan en la imagen.

- El bloque *Home Control* es responsable de recopilar información y controlar dispositivos.
- El bloque *Home automation* activa comandos basados en las configuraciones de usuario.
- El bloque *Smart Home* dispara comandos basados en comportamientos anteriores.

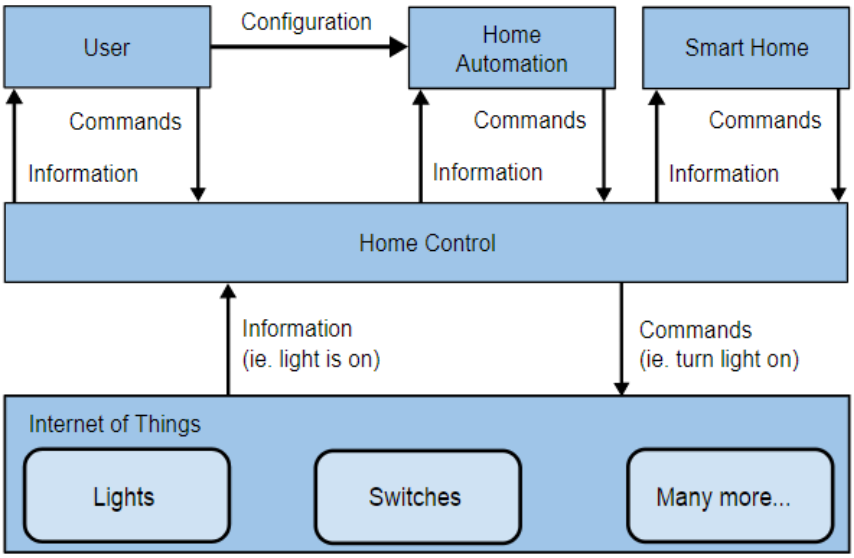


Figura 4-3. Arquitectura del hogar.

El núcleo de Home Assistant es responsable del control de la casa y contiene cuatro partes que hacen esto posible:

- ***Event Bus***: facilita la activación y la escucha de eventos. Es el corazón de Home Assistant.
- ***State Machine***: realiza un seguimiento de los estados de las cosas (conviene aclarar que llamo cosas a los componentes del sistema como luces, sensores, etc...) y dispara un evento *state_changed* cuando un estado ha sido cambiado.
- ***Service Registry***: escucha eventos en el bus de *call_service* y permite que otros códigos registren servicios.
- ***Timer***: envía un *time_changed* cada segundo en el bus de eventos.

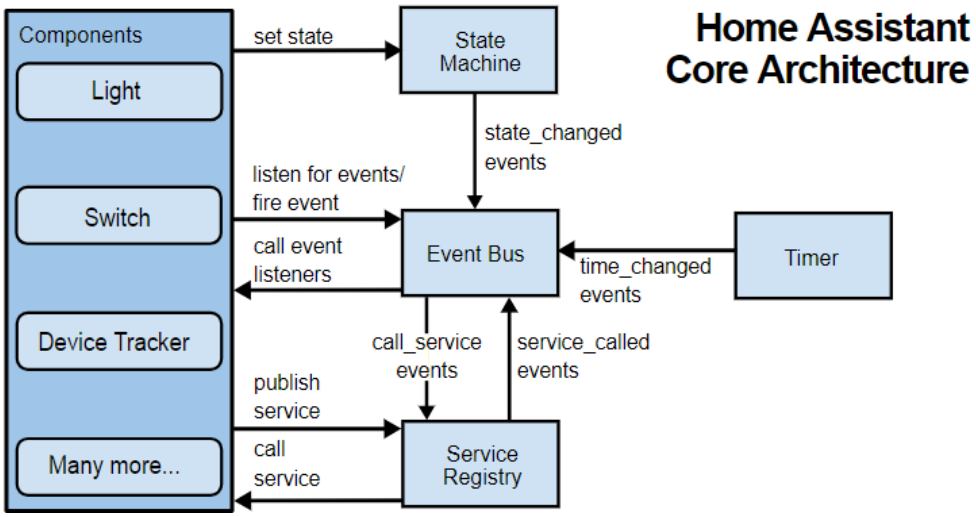


Figura 4-4. Arquitectura HA.

4.3.2 Arquitectura de componentes

Home Assistant se puede ampliar con componentes. Cada componente es responsable de un dominio específico dentro de Home Assistant. Los componentes pueden escuchar o desencadenar eventos, ofrecer servicios y mantener estados.

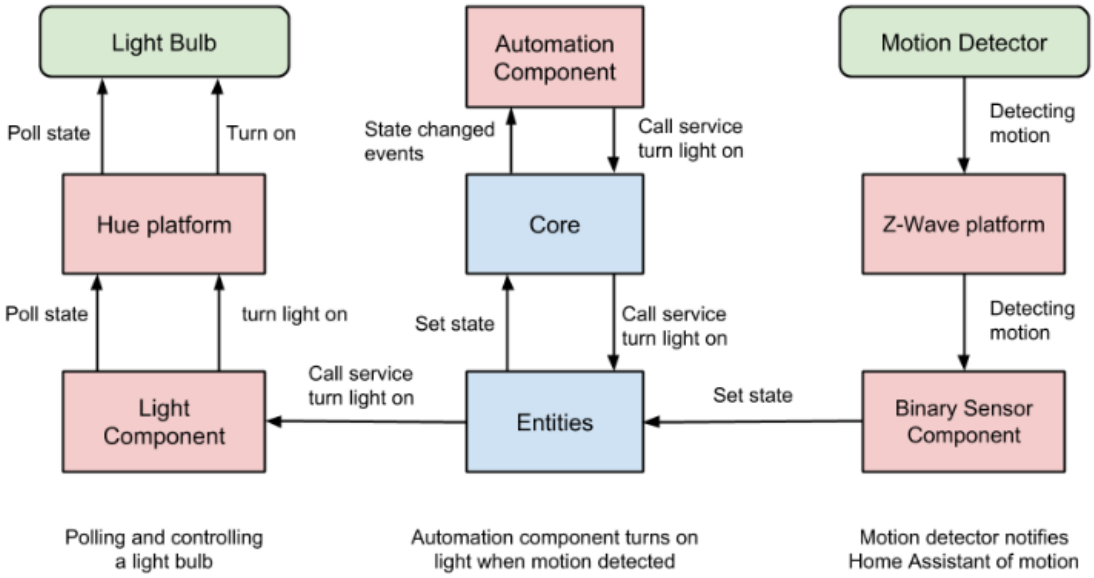


Figura 4-5. Arquitectura componentes.

Hay dos tipos de componentes dentro de Home Assistant: componentes que interactúan con un dominio de Internet de las Cosas, y componentes que responden a eventos que ocurren dentro de Home Assistant.

Componentes que interactúan con un dominio de Internet de las cosas.

Estos componentes rastrean los dispositivos dentro de un dominio específico y constan de una parte central y una lógica específica de la plataforma. Estos componentes hacen que su información esté disponible a través de *State Machine* y *Event Bus*. Los componentes también registran servicios en el registro de servicios para exponer el control de los dispositivos.

Por ejemplo, el componente "*switch*" incorporado es responsable de la interacción con diferentes tipos de interruptores. Una plataforma proporciona soporte para un tipo particular o marca de dispositivo. Por ejemplo, un conmutador podría usar una plataforma WeMo u Orvibo y una bombilla podría interactuar con la plataforma Hue o LIFX.

Componentes que responden a eventos que suceden dentro de Home Assistant.

Estos componentes proporcionan pequeñas piezas de lógica de automatización del hogar o implican servicios que realizan tareas comunes dentro del hogar.

Por ejemplo, el componente *device_sun_light_trigger* rastrea el estado de los dispositivos y el sol para asegurarse de que las luces están encendidas cuando oscurece y las personas están en casa. El componente usa lógica como esta:

En el caso de que el dispositivo del usuario cambie al estado 'Home':

Si el sol se ha puesto y las luces no están encendidas:

Enciende las luces

En el caso de que todos los dispositivos rastreados (usuarios de la casa, por ejemplo) cambien su estado a 'Not Home':

Si las luces están encendidas:

Apaga las luces

Una imagen completa de la arquitectura de Home Assistant haciendo unión de todas las piezas es la siguiente:

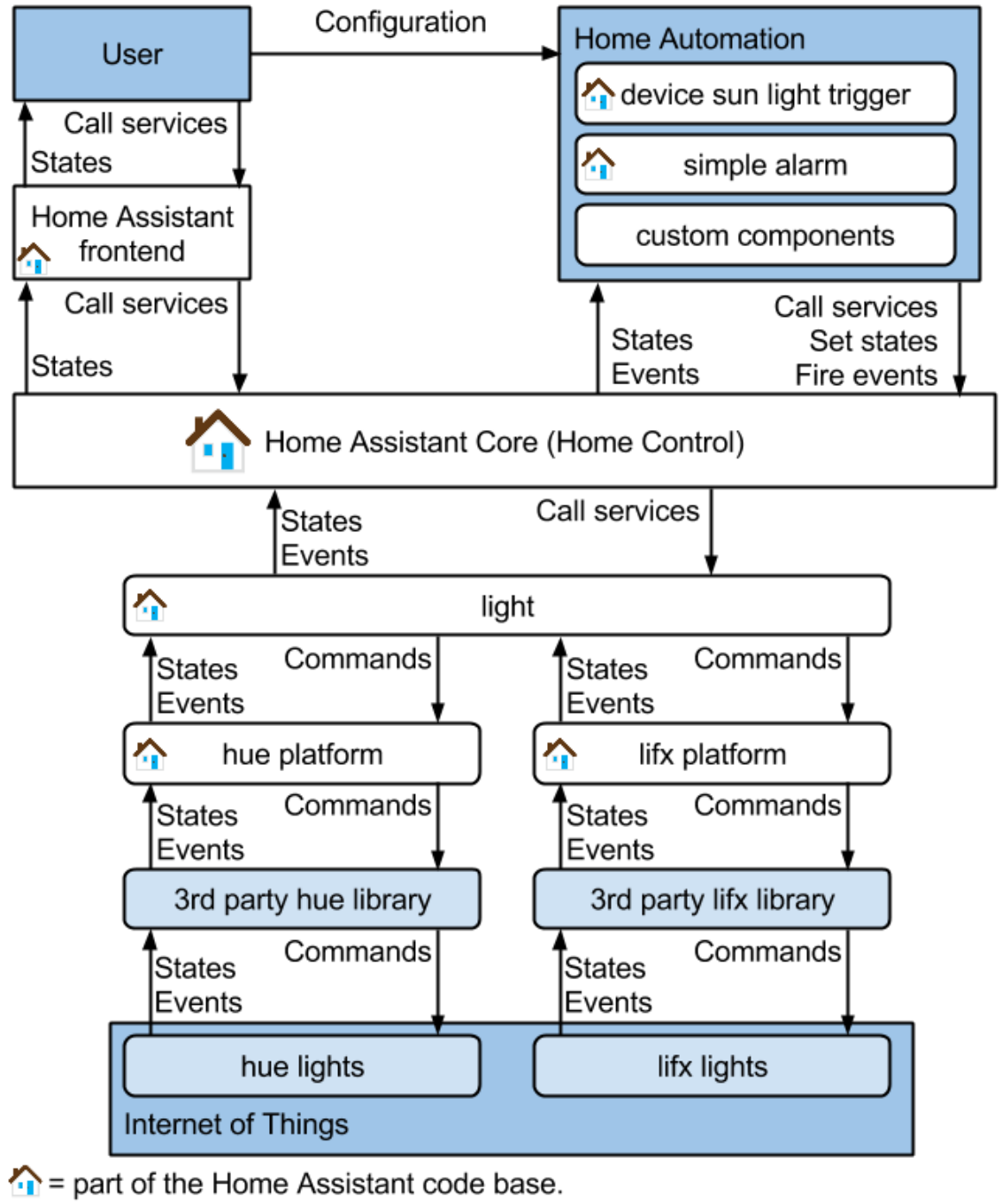


Figura 4-6. Arquitectura completa.

4.4 Entidades

Cada dispositivo está representado en Home Assistant como una entidad. Una entidad es la que proporciona los datos al sistema tal como un sensor o un interruptor. Como usuario, nuestra preocupación no debe ser cómo funcionan internamente. En su lugar existen unas clases de entidad que implementan las propiedades y métodos necesarios para el tipo de dispositivo que está integrando.

Es posible crear tus propias entidades, por lo que Home Assistant te da unas pautas para que sea posible la integración en su plataforma. Los proveedores de entidades se tienen que unir a estas reglas para que sus dispositivos sean compatibles con Home Assistant.

Existen varias estrategias para mantener la entidad sincronizada con el estado del dispositivo, la más popular es el *polling* (sondeo).

4.4.1 Polling

Con el sondeo, Home Assistant pide a la entidad de vez en cuando (dependiendo del intervalo de actualización del componente) que obtenga el estado más reciente. Home Assistant sondeará una entidad cuando la propiedad *should_poll* regrese *True* (el valor predeterminado). Se puede implementar la lógica de actualización utilizando *update()* o el método asíncrono *async_update()*. Este método debería obtener el último estado del dispositivo y almacenarlo en una variable de instancia para que las propiedades lo devuelvan.

El código es responsable de informar a Home Assistant que hay una actualización disponible. Hay que asegurar de que se tiene la propiedad *should_poll* a *False*.

Cada vez que reciba un nuevo estado de su suscripción, se puede decir a Home Assistant que hay una actualización disponible mediante una llamada a *schedule_update_ha_state()*o una devolución de llamada asíncrona *async_schedule_update_ha_state()*. Se puede el booleano *True* al método si desea que Home Assistant llame a su método de actualización antes de escribir la actualización en Home Assistant.

4.4.2 Propiedades genéricas

Nombre	Tipo	Valor por defecto	Descripción
assumed_state	boolean	False	Devuelve True si el estado se basa en nuestra suposición en lugar de leerlo desde el dispositivo.
Available	boolean	True	Indica si Home Assistant puede leer el estado y controlar el dispositivo subyacente.
device_state_attributes	dict	None	Información extra para almacenar sobre el estado. Debe ser información que explique el estado, no debe ser información estática como la versión de firmware.
entity_picture	URL	None	Url de una imagen para mostrar para la entidad.
name	string	None	Nombre de la entidad

should_poll	boolean	True	Para que Home Assistant sondee una entidad (con el valor true lo hace)
unique_id	string	None	Un identificador único por entidad. Necesita ser único dentro de una plataforma (es decir light.hue). No debe ser configurable por el usuario o ser modificable.

Tabla 4-1. Propiedades genéricas.

La clase base de la entidad tiene algunas propiedades que son comunes entre todas las entidades en Home Assistant. Estas se pueden incluir a cualquier entidad independientemente del tipo que sea (sensor, alarma, climatización ... etc). Todas estas propiedades son opcionales y no necesitan implementarse.

4.4.3 Propiedades avanzadas

Las siguientes propiedades también están disponibles en las entidades. Sin embargo, son solo para uso avanzado y deben usarse con precaución

Nombre	Tipo	Valor por defecto	Descripción
hidden	boolean	False	Indica si la entidad no debe mostrarse en la interfaz.
Icon	icono	None	Icono para usar en la interfaz. Los iconos comienzan con mdi:más un identificador . Probablemente no lo necesite ya que Home Assistant ya proporciona íconos predeterminados para todos los dispositivos.

Tabla 4-2. Propiedades avanzadas.

A continuación, se listará distintas entidades recogidas por Home Assistant

- Alarmas
- Sensores Binarios
- Climatizadores
- Luces
- Reproductores media
- Switch
- Aspiradoras
- … y un largo etc

Entrar en cada una de ellas haría de un documento muy extenso, y de muy poca utilidad, por lo que he decidido poner como ejemplo la entidad sensor.

4.4.4 Entidad sensor

Un sensor es una entidad de solo lectura que proporciona información. La información tiene un valor y, opcionalmente, una unidad de medida.

Nombre	Tipo	Valor por defecto	Descripción
Satate	string	required	El valor del estado del sensor
unit_of_measurement	string	none	La unidad de medida en la que el sensor está expresado
device_class	string	none	Tipo de sensor

Tabla 4-3. Propiedades sensor.

Existen muchas clases de sensores con distintas medidas. Si especifica una clase de dispositivo, la entidad de sensor deberá devolver la unidad de medida correcta.

Tipo	Unidad	Descripción
batería	%	% de batería que queda.
humedad	%	% de humedad en el aire.
iluminancia	lx / lm	Nivel de luz.
Intensidad de señal	dB / dBm	Intensidad de señal.
temperatura	° C / ° F	Temperatura.
marca de tiempo	ISO8601	Marca de tiempo
poder	W, kW	Poder.
presión	hPa, mbar	Presión.

Tabla 4-4. Distintas medidas.

4.5 Autenticación

Home Assistant tiene un sistema de autenticación incorporado que permite a diferentes usuarios interactuar con Home Assistant. El sistema de autenticación consta de varias partes.

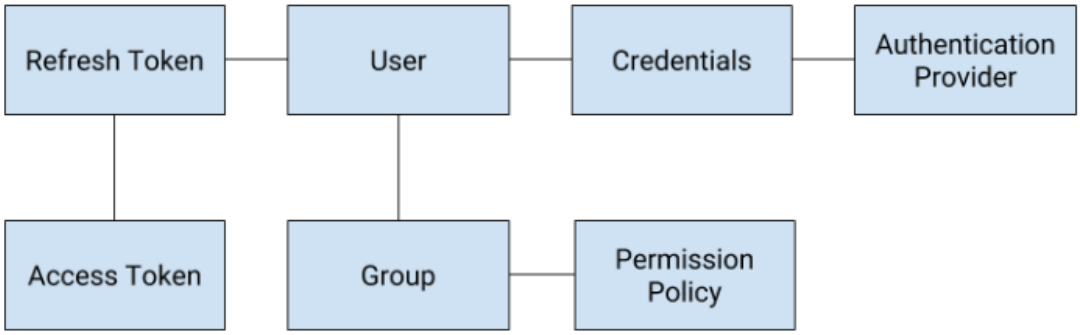


Figura 4-7. Autenticación.

4.5.1 Bloque Porveedor de Autenticación

Se utiliza un proveedor de autenticación para que los usuarios se autentifiquen a sí mismos. Depende del proveedor de autentificación elegir el método de autentificación y el backend a utilizar. De forma predeterminada, habilitamos el proveedor de autenticación Home Assistant incorporado que almacena a los usuarios de manera segura dentro de su directorio de configuración.

Los proveedores de autentificación que utilizará Home Assistant se especifican dentro del archivo *configuration.yaml*. Es posible tener activas varias instancias del mismo proveedor de autentificación y en ese caso, cada uno será identificado por un identificador único. Los proveedores de autentificación del mismo tipo no compartirán credenciales.

4.5.2 Bloque Credenciales

Las credenciales almacenan la autenticación de un usuario con un proveedor de autenticación específico y esto se produce cuando un usuario se autentica con éxito. Permitirá que el sistema encuentre al usuario. Si el usuario no existe, se creará un nuevo usuario. Este usuario no se activará, pero requerirá la aprobación del propietario.

Es posible que un usuario tenga múltiples credenciales vinculadas a él, pero sin embargo, solo puede tener una única credencial por proveedor de autenticación específico.

4.5.3 Bloque Usuario

Cada persona es un usuario en el sistema. Para iniciar sesión como un usuario específico, es necesario autetificarse con cualquiera de los proveedores de autenticación que están vinculados a este usuario. Cuando un usuario inicia sesión, recibirá una actualización y un token de acceso para realizar solicitudes a Home Assistant.

El usuario que se crea durante la incorporación se marcará como "propietario". El propietario puede administrar otros usuarios y siempre tendrá acceso a todos los permisos.

4.5.4 Bloque Grupos

Los usuarios son miembros de uno o más grupos. La pertenencia a un grupo es la forma en que un usuario recibe permisos.

4.5.5 Bloque Política de Permisos

Esta es la política de permisos que describe a qué recursos tiene acceso un grupo. Los permisos limitan las cosas que un usuario tiene acceso o puede controlar. Los permisos también se adjuntan a grupos, de los cuales un usuario puede ser miembro.

4.5.6 Bloque Acceder y Refrescar Tokens

Las aplicaciones que desean acceder a Home Assistant le pedirán al usuario que inicie un flujo de autorización. El flujo da como resultado un código de autorización cuando un usuario se autoriza con éxito en Home Assistant. Este código se puede utilizar para recuperar un acceso y un token de actualización. El token de acceso tendrá una duración limitada, mientras que los tokens de actualización seguirán siendo válidos hasta que un usuario lo elimine.

El token de acceso se utiliza para acceder a las API de Home Assistant. El token de actualización se utiliza para recuperar un nuevo token de acceso válido.

Hay tres tipos diferentes de tokens de actualización:

- Normal: estos son los tokens que se generan cuando un usuario autoriza una aplicación. La aplicación mantendrá estos tokens en nombre del usuario.
- Token de acceso de larga duración: son tokens de actualización que respaldan un token de acceso de larga duración. Se crean internamente y nunca se exponen al usuario.
- Sistema: estos tokens están limitados para ser generados y utilizados por usuarios del sistema como Hass.io. Nunca están expuestos al usuario.

5 HOME ASSISTANT A NIVEL DE USUARIO

En este punto se explicará home Assistant a nivel de usuario, y en él se podrá ver las distintas posibilidades que existen para su instalación, y además la configuración y uso de la plataforma.

5.1 Modos de instalación

Existe diferentes formas de instalar Home Assistant, pero antes cabe destacar que el hardware donde será alojado será una raspberry Pi 3 y los requisitos a nivel de hardware que se recomiendan usar son: un almacenamiento de 4 GB, una memoria de 1 GB, una conexión a red de 100Mb/s, una intensidad de 2.5A.

Existen otras posibilidades si se quieres más potencia de computación y poder utilizar el hardware para otros servicios como son los Intel NUC.

Intel NUC es un potente mini PC de cuatro por cuatro pulgadas con características de entretenimiento, juegos y productividad, que incluyen una placa personalizable lista para aceptar la memoria, el almacenamiento y los sistemas operativos que desee.

Figura 5-1. Intel NUC.

El software, HA propiamente dicho, tienen tres posibles formas de instalación que son Hass.io, Hassbian, y la instalación directa de HA en Raspbian.

5.1.1 Hass.io

Hass.io es la distribución creada por los desarrolladores del sistema que incluye el software Home Assistant en una distribución de Linux optimizada para funcionar de la mejor forma en un hardware específico bajo docker. Es decir, un sistema hecho a medida para funcionar de la mejor forma.

Por lo tanto Hass.io es el resultado de la combinación de Home Assistant y otras herramientas (hassio-supervisor) que permiten administrar las actualizaciones de HA y agregar soporte para complementos. Hass.io se distribuye a través de contenedores llamados docker: Home Assistant, hassio-supervisor y los distintos complementos se tienen en contenedores diferentes.

Home Assistant es el software en sí que gestiona todo nuestro sistema domótico, viene dentro de Hass.io y además, incluye un sistema de Add-Ons que hace mucho más fácil de instalar ciertas cosas, así como actualizarlas. Facilita un poco la vida a la hora de mantener el sistema.

Gracias a hassio-supervisor tenemos la capacidad de actualizar Home Assistant, administrar complementos y crear copias de seguridad de toda la instalación con un solo clic a través de la pestaña "Hass.io" ubicada en el panel izquierdo de la interfaz gráfica. Pestaña que no está presente en una instalación "independiente" de Home Assistant

Hass.io es una forma muy sencilla de utilizar la plataforma domótica, pero tiene la desventaja de que tienes menos control en la manera en que la puedes hacer configuraciones manuales.

Docker es una herramienta para la distribución y gestión de contenedores. Un contenedor es una colección de uno o más procesos aislados del resto del sistema, todos los archivos necesarios para realizar estos procesos se distribuyen en una imagen. Docker podría verse como un sistema de virtualización, pero la diferencia es que una máquina virtual emula un sistema operativo completo (incluido el hardware), Docker comparte el mismo kernel (que es el kernel de Linux) entre todos los contenedores para no tener la pérdida de rendimiento que tendría con la virtualización

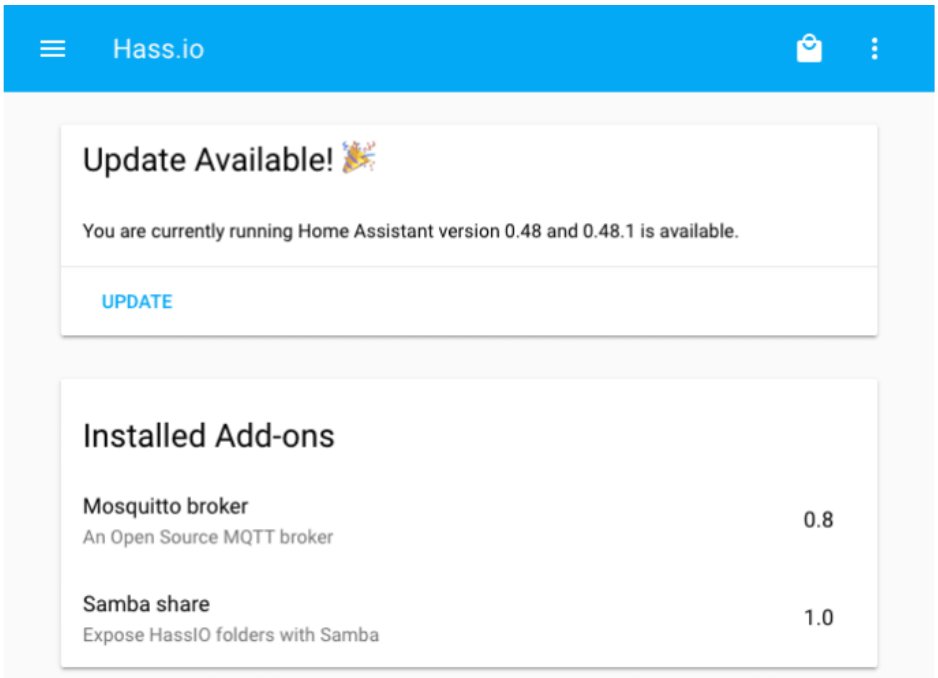


Figura 5-2. Interfaz Hass.io.

5.1.2 Hassbian

Hassbian es un sistema operativo que, a diferencia de Hass.io, se basa en Raspbian (que a su vez se basa en Debian) y se distribuye con Home Assistant instalado de fábrica, que en este caso no se ejecuta en un contenedor Docker, sino directamente en el sistema operativo en un entorno virtual de Python 3.

Está hecha para aquellos que desean una experiencia Linux más tradicional y tienen experiencia con Linux, o que tienen la intención de aprender como es mi caso.

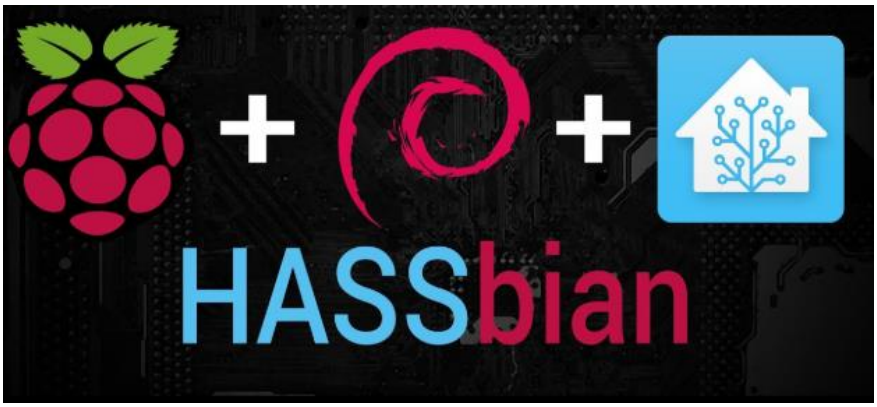


Figura 5-3. Hassbian.

5.1.3 Home Assistant

Home Assistant es en sí mismo el software para la domótica. Está escrito en Python y es de código abierto, por lo tanto, se puede instalar en cualquier plataforma compatible con Python y es gratuito.

A modo de resumen nos quedamos con estos conceptos; **Home Assistant** es el software que todos conocemos; **Hass.io** es un método de instalación de Home Assistant que ofrece herramientas adicionales para administrar actualizaciones de HA, copias de seguridad y complementos, se basa en docker y está optimizado para dispositivos integrados (como Raspberry Pi); **Hassbian** es un sistema operativo completo, basado en Raspbian, que tiene un Home Assistant preinstalado.

5.2 Configuración Home Assistant

Una vez que hemos instalado HA de alguna de las distintas formas descritas arriba (la manera de instalación se describirá en el punto 6.1) vamos a encargarnos de entender y configurar por dentro todo el potencial que nos ofrece la plataforma.

Home Assistant se instala en un entorno virtual en la ruta */srv/homeassistant*. Se inicia como un servicio ejecutado por el usuario *homeassistant* y su configuración se encuentra localizada en la dirección */home/homeassistant/ .homeassistant*. Con esto se asegura que las instalaciones de Python y de Home Assistant no interfieren entre sí.

Para iniciar sesión en su Raspberry Pi con Hassbian, se utilizará un cliente ssh. Dependiendo de dónde queramos gestionar la plataforma existen diferentes formas para usar el cliente ssh, por ejemplo, en Linux y Mac OS generalmente tienen un cliente ssh instalado. Los usuarios de Windows pueden utilizar clientes ssh como el software Putty .

El nombre de usuario predeterminado para conectarse a la raspberry es *pi* y la contraseña es *raspberry*. Los usuarios de Linux y Mac OS ejecutarán el siguiente comando en un terminal.

```
ssh pi@dirección-ip-de-pi
```

Los usuarios de Windows inician Putty deberán ingresar la dirección IP de la Raspberry Pi en el campo *nombre de host* y el puerto 22 en el campo *puerto*. Luego se deberá hace click en *Abrir* y se abrirá una ventana de terminal. Se introducirá las credenciales, nombre de usuario predeterminado que es *pi* y la contraseña es *raspberry*.

Para iniciar HA una vez iniciada la sesión y establecida la conexión con la Raspberry se ejecutará el siguiente comando

```
sudo systemctl start home-assistant@homeassistant.service
```

De la misma forma para detener o reiniciar el servicio se tendrá que ejecutar los siguientes comandos

```
sudo systemctl stop home-assistant@homeassistant.service
sudo systemctl restart home-assistant@homeassistant.service
```

Para empezar a poner en marcha todos los componentes, automatizaciones y configurar la instalación, se tendrá que trabajar con los archivos de configuración de Home Assistant. Cuando se inicia por primera vez, HA escribe un archivo de configuración predeterminado que habilita la interfaz web y el descubrimiento de dispositivos que estén en la red de casa.

Para poder acceder a la interfaz web se deberá acceder a http://dirección-ip:8123/: por ejemplo, si nuestro sistema tiene la dirección 192.168.08.48, se econtrará la interfaz web de Home Assistant en la diección como http://192.168.08.48:8123/.

Dentro de la carpeta de configuración está el fichero más importante, con el que se trabaja y el que se crea cuando se inicia por primera vez HA. Este fichero recibe el nombre de *configuration.yaml*. Este es el archivo principal que contiene las integraciones que se cargarán con sus configuraciones.

Home Assistant utiliza el formato **YAML** que significa "*Another Another Markup Language* "y, al igual que Python, utiliza espacios en blanco (no tabulaciones) para delimitar las secciones de código.

5.2.1 YAML

YAML es un lenguaje de serialización de datos diseñado para ser leído y escrito por humanos. Normalmente se utiliza para definir archivos de configuración, aunque también es posible serializar objetos, es decir, escribir la estructura de un objeto en modo cadena de texto para posteriormente poderlo recuperar. A diferencia de Python, YAML no permite tabulaciones literales. Este lenguaje es muy legible para las personas, más legible que JSON y sobretodo que XML. Es case sensitive y normalmente lo encontraremos con la extensión .yml o incluso .yaml.

Los archivos YAML comienzan con una lista. Cada elemento de la lista es una lista de pares clave / valor, comúnmente llamados "hash" o "diccionario". Por lo tanto, necesitamos saber cómo escribir listas y diccionarios en YAML.

Existen unas reglas generales que deben cumplirse en un documento YAML:

- Los datos de un documento YAML deben ser legibles, imprimibles y utilizando caracteres Unicode, UTF-8 o UTF-16.
- Los comentarios se realizan utilizando el carácter '#' dentro de la línea que contiene el comentario.
- Los caracteres ',' y ';' deben ir seguidos de un espacio en blanco. De esta forma, se podrán representar valores que queramos que tengan esos caracteres.
- Los espacios en blanco están permitidos, pero no los tabuladores.
- Las listas comienzan por el carácter '-' con un valor por cada línea, aunque también se pueden utilizar corchetes [] poniendo los valores dentro de ellos separados por comas ',' junto con un espacio en blanco.
- Un vector estará formado por el par clave/valor, estando separados ambos por ':' poniendo uno por línea, aunque también podemos utilizar {} poniendo cada uno de ellos dentro separados por comas ',' junto con un espacio en blanco.
- Se pueden utilizar caracteres de escape '\' para representar caracteres especiales.
- Para incluir múltiples documentos, los separaremos por tres guiones seguidos '—' e indicando el fin de un documento con tres puntos seguidos …
- La estructura del documento se denota indentando con espacios en blanco; sin embargo, no se permite el uso de caracteres de tabulación para indentar.

A continuación, se expondrán una serie de ejemplos para entender mejor YAML

1. Listas

```
# Libros favoritos, formato de bloque
```

```
 - HarryPotter
 - Enigma
 - LaFortaleza
 - ElPrincipito

# Libros favoritos, formato en línea
[HarryPotter, Enigma, LaFortaleza]
[TusZonasErroneas, Elprincipito]
```

2. Vectores asociativos

```
#En formato bloque
nombre: Santiago Romero
edad: 24

#En formato línea
{nombre:Santiago Romero, edad: 24}
```

3. Lista de vectores asociativos

```
- {nombre: Santi Romero, edad: 4}
- nombre: Jose Luis
  edad: 28
```

4. Vectores asociativos de listas

```
hombres: [Santi Romero, José Luis]
mujeres:
 - Marisa Cabrera
 - Veronica Fernandez
```

También han de mencionarse los diccionarios que permiten guardar un conjunto no ordenado de pares clave-valor (key: value)

```
# An employee record
martin:
    name: Martin D'vloper
    job: Developer
    skill: Elite
```

Son posibles estructuras de datos más complicadas, como listas de diccionarios, diccionarios cuyos valores son listas o una combinación de ambos

```
# Empleados
-  martin:
    nombre: Martin Gutierrez
    trabajo: programador
    habilidades:
      - python
      - perl
     - pascal
-  Pepe:
    name: Pepe Gozalez
    trabajo: programador
```

```
habilidades:
  - lisp
  - fortran
  - erlang
```

Aprovechando que en el grado hemos estudiando JSON y es un lenguaje bastante extenso y conocido, haremos una comparativa para poder verlo con más claridad

Así, por ejemplo, el siguiente JSON:

```
{ "configuracion": {
      "IPs": [ "127.0.0.1",
               "192.168.0.1"
             ],
      "puerto": 8000,
      "nombre": "miservidor.com"
   }
}
```

Se escribiría así en YAML:

```
configuracion:
   IPs:
     - 127.0.0.1
     - 192.168.0.1
   puerto: 8000
   nombre: miservidor.com
```

Por lo tanto, para cada integración que desee utilizar en Home Assistant, se debe agregar código en el archivo *configuration.yaml* para especificar su configuración. En el siguiente ejemplo (ya es un ejemplo de HA) se muestra la manera de incluir un componente de notificación con la más que conocida plataforma telegram.

```
notify:
  - name: telegramsanti
    platform: telegram
    chat_id: 556769163
```

`Notify` permite enviar notificaciones a una amplia variedad de plataformas como por ejemplo pushbullet que es la encargada de enviarnos la notificación al teléfono.

El **componente** proporciona la lógica central para algunas funciones (`notify` se encarga del envío de notificaciones).

La **plataforma** realiza la conexión a una plataforma de software o hardware específica (*telegram*).

Los conceptos básicos de la sintaxis YAML definidos por los desarrolladores de Home Assistant son colecciones de bloques y asignaciones que contienen pares clave-valor. Cada elemento de una colección comienza con un '-' (lo denominado listas) mientras que las asignaciones tienen el formato *key: value*. Si se utilizan claves (*key*) duplicadas, se utiliza el último valor de una clave. Se debe tener en cuenta que los espacios delimitan las anidaciones, y esto quiere decir que por ejemplo `platform: telegram` es una propiedad del componente `notify` ya que está anidada dentro.

Existen webs y herramientas que validan tú codigo YAML online y poder corregirlo antes de cargarlo en HA. YAMLint es uno de ellos, aunque en la propia interfaz de HA existe una herramienta validadora.

A continuación, se definirá otro componente importante dentro del *configuration.yaml* como es un

`input_select`. Es un seleccionador de entradas que permite al usuario definir una lista de valores que se pueden seleccionar a través de la interfaz y se puede usar para la automatización. Cuando un usuario selecciona un nuevo elemento del `input_select`, se genera un evento de transición de estado que puede ser usado en una automatización.

La estructura de un `input_select` es la siguiente:

`input_select`

(mapa) (Requerido) Alias para la entrada. Se permiten múltiples entradas.

`options`

(lista) (Requerido) Lista de opciones para elegir.

`name`

(cadena) (Opcional) Nombre descriptivo de la entrada.

`initial`

(mapa) (Opcional) Valor inicial cuando se inicia Home Assistant.

Valor por defecto: primer elemento de opciones

`icon`

(icono) (Opcional) Icono para mostrar para el componente.

A continuación, se muestra un ejemplo en el `configuration.yaml`

```
#Ejemplo en el configuration.yaml
input_select:
  quien_limpia:
    name: quien limpia hoy
    options:
      - Santi
      - Emilio
    initial: Emilio
    icon: mdi:chef-hat
  habitación_invitados:
    options:
      - Visitantes
      - Visitantes con niños
      - Habitacion vacia
```

Vemos como se usa lo descrito arriba, una colección de bloques para el valor de las opciones y las otras propiedades como el nombre se especifican mediante asignaciones. Aquí existen dos entradas para `input_select` que son `quien_cocina` o `habitacion_invitados` son los nombres del `input_select` y los valores los que están anidados debajo de ambos.

En el siguiente ejemplo se mostrará cómo anidar una colección de asignaciones en una asignación. Vamos integrar varios sensores diferentes, tal y como vemos en el `state_topic`.

```
sensor:
  - platform: mqtt
    state_topic: sensor/topic
  - platform: mqtt
    state_topic: sensor2/topic
```

Para mejorar la legibilidad del código del archivo de configuración principal, se recomienda separar en distintos

archivos YAML en función de los componentes declarados, con una sintaxis del tipo *lights: !include lights.yaml.* Esta declaración le dice a HA que inserte el contenido de lights.yam en ese punto. Hay que tener en cuenta que solo puede haber un *!include* para cada componente.

Otra de las opciones que hay será la de incluir uno o varios *ficheros .yaml* dentro de una carpeta del directorio principal de homeassistant.

Otro aspecto a destacar es que HA distingue entre mayúsculas y minúsculas, un estado de 'on' no es el mismo que 'On' o 'ON'.

5.2.2 Variables de configuracion inicial

Inicialmente HA intentará reconocer las variables de configuración incial básicas como la geolocalización del usuario y esto lo hace mediante la dirección IP. Asigna una latitud, longitud, elevación, unidad de temperatura, unidad de medida (métrica), zona horaria por defecto. Esto se puede cambiar modificando en el *configuration.yaml* los siguientes parámetros.

Variables de configuración inicial

`Latitude`

(float)(Optional) Latitud de tu localización. Fundamental para conocer la salida y puesta de sol, entre otros ejemplos.

`longitude`

(float)(Optional) Longitud de tu localización. Fundamental para conocer la salida y puesta de sol, entre otros ejemplos.

`elevation`

(integer)(Optional) Atura por encima del mar.

`unit_system`

(string)(Optional) metric para el sistema métrico, imperial para el sistema imperial (el que usa yardas, pies, pulgadas, etc).

`time_zone`

(string)(Optional) Para elegir la zona horaria nos debemos didrigir a la columna “TZ database name” de Wikipedia https://en.wikipedia.org/wiki/List_of_tz_database_time_zones. Nuesta zona horaria es según la columna *Europe/Madrid*

`name`

(string)(Optional) Nombre de la localización dónde se está ejecutando HA

A continuación, se muestra un ejemplo:

```
homeassistant:
   latitude: 40.2970592
   longitude: -4.5882800
   elevation: 659
   unit_system: metric
   time_zone: Europe/Madrid
   name: Santi Home
```

5.3 Añadir dispositivos a Home Assistant

Cada dispositivo inteligente consiste en el dispositivo en sí mismo y la pieza que lo hace inteligente, la conectividad. La conectividad de un dispositivo puede consistir en control, estado o ambos. El estado describe lo que un dispositivo está haciendo en este momento. Por ejemplo, una luz puede estar encendida con un color rojo y un brillo con cierto valor. El control consiste en comunicarse con el dispositivo inteligente mediante el envío de comandos a través de una API.

El estado se puede dividir en 5 categorías las cuales no se excluyen mutuamente: un estado del dispositivo puede estar disponible a través de la nube y la conectividad local

- Estado asumido

  No se puede obtener el estado del dispositivo y lo que se hace es asumir el estado basado en el último comando. Estos son dispositivos que únicamente permiten ser controlados y que no tienen las capacidades para hacer que su estado esté disponible. Por ejemplo, los dispositivos con controles remotos de infrarrojos como televisores y AC. Se presionar el botón de encendido en el mando, pero solo se puede asumir que el comando fue recibido y ejecutado con éxito

  La automatización del hogar tendrá que acercarse a dichos dispositivos basándose en la suposición de que los comandos se reciben correctamente: utilizando actualizaciones optimistas.

- Encuestando a la nube (Cloud Polling)

  La integración de este dispositivo se realiza a través de la nube y requiere una conexión a Internet activa. Estos son dispositivos que solo informarán su estado a su propio backend de la nube. El backend de la nube permitirá leer el estado, pero no notificará cuando haya llegado un nuevo estado. Esto requiere que la automatización del hogar verifique con frecuencia si el estado se ha actualizado.

- Empuje de la nube (Cloud Push)

  Todo de la sección anterior se aplica a este. Además, la nube también notificará a automatización del hogar cuando haya llegado un nuevo estado.

- Encuesta local (Local Pollig)

  Estos dispositivos ofrecerán una API que es accesible localmente. La automatización del hogar deberá verificar con frecuencia si el estado se ha actualizado. Además, no depende de internet, solo conexión con el dipositivo.

- Empuje local (Local Push)

  Ofrece comunicación directa con el dispositivo. Home Assistant será notificado tan pronto como haya un nuevo estado disponible.

  El control de un dispositivo se puede hacer, como el estado, a través de la nube y / o la conectividad local. Pero la parte más importante del control es saber si el comando enviado al dispositivo fue un éxito y conocer el nuevo estado del dispositivo.

- No hay control

  Estos dispositivos no pueden ser controlados y sólo ofrecerán estado.

- Estado de la encuesta después de enviar el comando

  Estos dispositivos requerirán que el estado sea sondeado después de enviar un comando para ver si un comando fue exitoso.

- El dispositivo empuja actualización de estado

  Estos dispositivos no devolverán un nuevo estado como resultado del comando, sino que impulsarán un nuevo estado de inmediato.

- El comando devuelve el nuevo estado

Estos dispositivos responderán al comando con el nuevo estado después de ejecutar el comando.

Home Assistant tiene un componente de descubrimiento automático con el que puede descubrir automáticamente muchos dispositivos y servicios disponibles en su red siempre y cuando se tenga escrito en el archivo de configuración.

El componente de descubrimiento recibe el nombre de `discovery` y para cargar este componente es necesario añadir estas líneas al *configuration.yaml.*

```
discovery:
  ignore:
    - sonos
    - samsung_tv
  enable:
    - homekit
```

En este ejemplo al componente de descubrimiento le estamos dando órdenes de que ignore la televisión de Samsung y dispositivos del fabricante 'Sonos'.

Variables de configuración

`Ignore`

(list)(Optional) Una lista de componentes que nunca serán configuradas automáticamente.

`Enable`

(list)(Optional) Una lista de plataformas no habilitadas por defecto que discovery debería descubrir.

Actualmente Discovery puede detectar una amplia gama de dispositivos entre los que destaco

- Apple TV
- Interruptores Belkin WeMo
- Google Cast
- Impresora Samsung SyncThru
- Televisores Samsung
- Puerta de enlace Xiaomi (Aqara)
- Lámparas y bombillas Yeelight
- …

Cada entidad necesita su propia entrada en el archivo *configuration.yaml*. Hay dos formas de inclusión para entradas múltiples

FORMA 1: Recoger todas las entidades bajo un "padre"

En este ejemplo se incluyen tres sensores, dos de ellos funcionando vía MQTT en distintas partes de la casa (cocina, dormitorio) y un sensor de tiempo vía web. Estos se recogen bajo la entidad padre `sensor`. También se incluye otra entidad `switch` con la plataforma "*vera*" que es un fabricante que crea componentes domóticos.

```
sensor:
  - platform: mqtt
    state_topic: "home/bedroom/temperature"
    name: "MQTT Sensor 1"
  - platform: mqtt
    state_topic: "home/kitchen/temperature"
    name: "MQTT Sensor 2"
  - platform: rest
```

```
    resource: http://IP_ADDRESS/ENDPOINT
    name: "Weather"

switch:
  - platform: vera
```

La forma 1 es la manera elegida en mi sistema domótico de inlcuir componentes, como se podrá ver en la inclusión de sensores.

FORMA 2: Listar cada dispositivo por separado

Debe agregar números o cadenas para diferenciar las entradas de las entidades, por ejemplo con los sensores anteriores se les pone la cadena *bedroom* o *kitchen.* El número o cadena adjunta debe ser único.

```
sensor bedroom:
  platform: mqtt
  state_topic: "home/bedroom/temperature"
  name: "MQTT Sensor 1"

sensor kitchen:
  platform: mqtt
  state_topic: "home/kitchen/temperature"
  name: "MQTT Sensor 2"

sensor weather:
  platform: rest
  resource: http://IP_ADDRESS/ENDPOINT
  name: "Weather"

switch 1:
  platform: vera

switch 2:
  platform: vera
```

5.3.1 Grupos

A medida que se van agregando componentes, la interfaz web se empieza a desordenar ya que aparecen distintos elementos dispersos por toda la página. Para evitar esto, se pueden utilizar grupos y vistas para limpiar la interfaz y colocar elementos relacionados en una determinada pestaña.

Un *group* es simplemente un objeto que contiene una lista de entidades. Visualmente, un *group* se representa como un panel o una ficha. Los grupos normalmente contienen una lista de entidades. Cada grupo consta de un nombre y una lista de con las ID de las entidades a agrupar. La ID de la entidades se pueden consultar en la pestaña herramientas de desarrollo de la interfaz web. En la imagen se marca en rojo la pestaña.

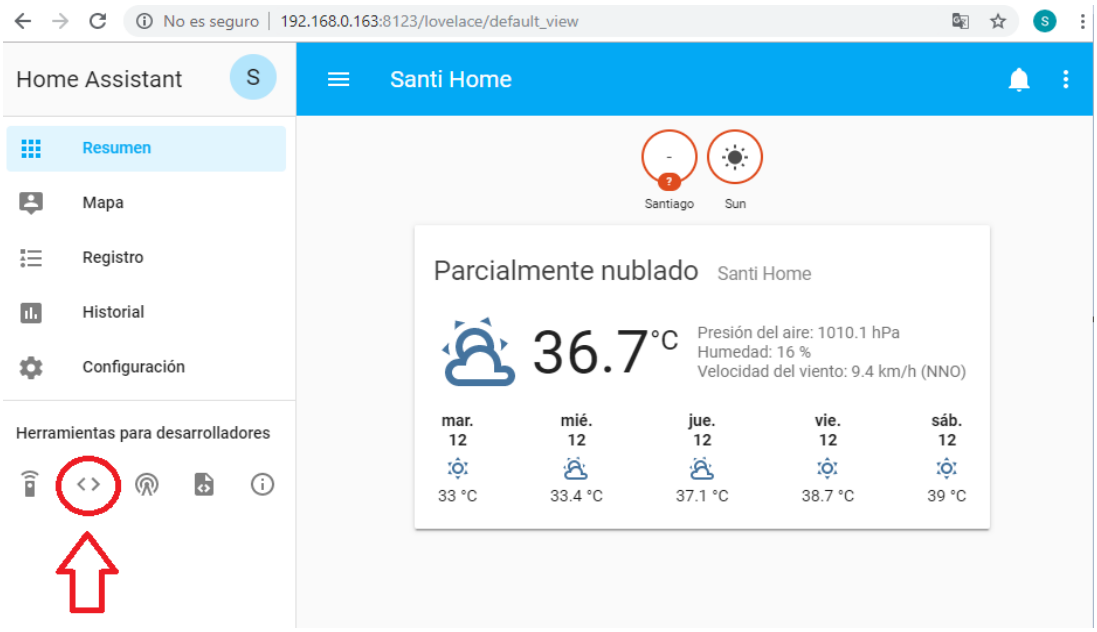


Figura 5-4. Grupos.

Variables de configuración

`name`

(string)(Optional) Nombre del grupo

`entities`

(list)(Required) Lista de entidades a agrupar

`icon`

(string)(Optional) El icono que se muestra en la web de HA

Un ejemplo en el fichero de *configuración.yaml* de grupo podría ser el siguiente

```
salon:
  name: Salón
  view: false
  entities:
    - light.salon
    - switch.sonoff_10008146b5
    - switch.07200091bcddc20d319a
    - sensor.dht_sensor_temperature
    - sensor.dht_sensor_humidity
```

Por último, cabe destacar que algunas integraciones crean automáticamente grupos especiales que contienen entidades de integración.

- group.all_switches
- group.all_lights
- group.all_devices (almacena por defecto los elementos descubiertos por la plataforma)
- group.all_scripts
- group.all_automations

Se puede ver la lista de grupos en la pestaña herramientas de desarrollador de la interfaz web.

5.3.2 Modificar entidades

En este apartado se mostrará cómo cambiar algunos parámetros de las entidades. En primer lugar, se detallan los puntos para poder cambiar el ID de la entidad *entity_id* con el que nos podemos referir a ella tal y como vimos en el apartado 5.3.1. Se puede utilizar la interfaz de usuario y los pasos a seguir son los siguientes:

- Selecciona entidad haciendo clic al 
- Clickea el engranaje en la esquina derecha
- Introducir nuevo nombre a la entidad
- Guardar

A continuación, se muestran las variables de configuración existentes para la modificación de entidades y por lo tanto llevar a cabo una personalización de la interfaz de usuario.

Variables de configuración

`friendly_name`

(string)(Optional) Nombre de la interfaz que aparece en la interfaz de usuario.

`homebridge_name`

(string)(Optional) Nombre que tienen la entidad en HomeBridge de Apple. Esto solo se utiliza si se hacen integraciones con esta marca.

`hidden`

(boolean)(Optional) Si se cambia el valor a true la entidad permanece oculta.

El valor por defecto es true.

`homebridge_hidden`

(boolean)(Optional) Si el valor se estable a true se oculta la entidad de HomeBridge

`entity_picture`

(string)(Optional) Se proporciona una URL que servirá como foto de la entidad

`icon`

(string)(Optional) Se pueden añadir iconos mediante mdi: (nombre icono). Este nombre del icono lo encontraremos en la siguiente dirección https://cdn.materialdesignicons.com/3.5.95/

`assumed_state`

(boolean)(Optional) Para los interruptores con estados asumidos son dos botones lo que se muestran, encender o apagar, en lugar del icono del interruptor (switch). Por lo que si queremos el icono por defecto es necesario ponder assumed_state a false. El valor por defecto es true.

`device_class`

(device_class)(Optional) Establece la clase del dispositivo, cambiando el estado y el icono del

dispositivo que se muestra en la interfaz de usuario No establece la unidad de medida.

Los tres tipos que soporta actualmente (device_class) son:

- Binary Sensor
- Sensor
- Cover (para persianas, toldos, etc)

`unit_of_measurement`

(string)(Optional) Define la unidad de medida.

`initial_state`

(string)(Optional) Establece el estado ninicial de las automatizaciones en on o off.

Para llevar a cabo la personalización de las entidades existen dos posibilidades

1. Mediante la interfaz de usuario

   En la interfaz de usuario podemos ver el menú *Configuración* y dentro el apartado *Personalización.* Cuando se selecciona una entidad se podrán ver todos los atributos existentes y se podrán modificar.

2. De forma manual en el *configuration.yaml*

   Dentro de `homeassistant`: se implementa `customize`, `customize_domain` o `customize_glob`

A continuación, se muestran ejemplos:

Customize: Se añade una entrada para cada entidad que se quiera personalizar

```
light.yeelight_color1_04cf8ca2deda:
    friendly_name: Luz Habitación
switch.07200091bcddc20d319a:
    friendly_name: Enchufe Salon
switch.sonoff_10008146b5:
    friendly_name: Luz cocina
    icon: mdi:lightbulb
```

Customize_domain: Personaliza todas las entidades bajo un dominio

```
customize_domain:
     light:
         icon: mdi:home
     automation:
         initial_state: 'on'
```

Customize_glob: Personalizar entidades que coincidan con un patrón

```
custostomize_glob:
      "light.kitchen_*":
                icon: mdi:description

      "scene.month_*_colors":
                hidden: true
                emulated_hue_hidden: false
                homebridge_hidden: true
```

HA ofrece un servicio para recargar las configuraciones sin tener que reiniciarlo. Para ello hay que dirigirse a las herramientas de desarrolladores, seleccionar el servicio *homeassistant.reload_core_config* y hacer clic en *servicio de llamadas*.

5.4 Automatizaciones

Las automatizaciones se dividen en tres partes: disparador (*trigger*), una condición (*condition*), y acción

(*action*). Ya se expuso en el punto 4.2 de este documento la explicación de cada parte de una automatización.

Para el estudio de las automatizaciones atendamos al siguiente ejemplo, "*A las siete de la mañana, de lunes a viernes, enciende la luz de la cocina*". Esta es una automatización que tengo en mi sistema como se podrá ver más adelante.

Trigger → A las siete de la mañana.

Los disparadores *triggers* son los que inician el procesamiento de una regla de automatización. Es posible especificar varios *triggers* para la misma regla y cuando cualquiera de los *triggers* se convierta en verdadero, se iniciará la automatización. Una vez que se inicia un *trigger*, Home Assistant validará las condiciones *condition*, si las hubiera, y activará la acción *action*. En este caso, cuando sean las siete de la mañana (07:00), es decir, el disparador checkea el tiempo.

Condition → De lunes a viernes.

Las condiciones son pruebas opcionales que pueden limitar una regla de automatización para que solo funcione en sus casos de uso específicos. Una condición prueba un estado actual del sistema, por ejemplo, si estamos entre semana.

Action → Enciende las luces de casa

La acción se realiza cuando se activa un disparador y se cumplen todas las condiciones. Enciende las luces es un ejemplo de acción.

Las reglas de automatización interactúan directamente con el estado interno de Home Assistant, es decir con los estados, los cuales se muestran a través de las herramientas del desarrollador. Estos están disponibles en la parte inferior de la barra lateral en la interfaz. El icono  mostrará todos los estados disponibles actualmente. Una entidad puede ser cualquier cosa, una luz, un interruptor, una persona e incluso el sol.

Tres parámetros importantes son:

- **ID_entidad:** es una referencia que asignamos a la entiedad. Ejemplo: *light.salon*
- **Estado:** el estado actual del dispositivo. Ejemplo: *home,* cuándo el dispotitivo de santi está en casa.
- **Atributos:** datos extras relacionados con el dispositivo y/o el estado actual. Ejemplo: *brigtness,* para definir la luminosidad de la luz.

Los cambios de estado se pueden utilizar como *triggers* y el estado actual se puede usar en *conditions*.

Las acciones llaman a los servicios, y para ver los servicios disponibles nos vamos a la herramienta de desarrollo de servicios  . Cada servicio tiene un dominio y un nombre. Por ejemplo, el servicio *light.turn_on* es capaz de encender cualquier luz en su sistema. Los servicios pueden pasar parámetros para, por ejemplo, indicar qué dispositivo activar o qué color usar.

Cuando se crea una nueva automatización, se habilitará a menos se agregue explícitamente *initial_state: false* o la desactive manualmente a través de la interfaz de usuario.

Se van a exponer como ejemplo dos automatizaciones, la primera que tiene como alias "enciende las luces" y la segunda "apaga las luces"

Debemos introducir `automation` dentro del configuration.yaml.

```
automation:
# Primera automatizacion
  - alias: 'enciende las luces'
    trigger:
      - platform: sun
        event: sunset
        offset: '-01:00:00'
      - platform: state
        entity_id: group.all_devices
        to: 'home'
    condition:
```

```
      - condition: time
        after: '16:00:00'
        before: '23:00:00'
    action:
      service: homeassistant.turn_on
      entity_id: group.living_room

# Segunda automatizacion
  - alias: 'apaga las luces'
    trigger:
      platform: state
      entity_id: group.all_devices
      to: 'not_home'
    action:
      service: light.turn_off
      entity_id: group.all_lights
```

La primera automatización recibe el nombre de 'enciende las luces' y su esquema es sencillo. Tiene dos disparadores que son la caída del sol detectada por HA con el componente que le viene integrado por defecto, y el otro disparador es que todos los dispositivos integrados pasen al estado 'home'. También existe una condición que es temporal, es decir para que ocurra debe ser después de las 16.00 y antes de las 23.00. Si disparador y condición se cumple se llamará a la acción.

La segunda automatización tiene como disparador que el grupo de todos los dispositivos de casa pasen al estado '*not_home*'. Si esto ocurre se llama a la acción de apagar todas las luces. No tiene condición.

También desde la interfaz de usuario de podrán configurar automatizaciones. Elegir *Configuración*, que se encuentra en la barra lateral, luego haga clic en *Automatización* para ir al editor de automatización y presionar el signo + en la esquina inferior derecha para comenzar con la automatización

Por último es importante comentar que para no hacer engorroso el archivo de configuración se podrá tener un archivo llamado *automations.yaml* que contendrá las automatizaciones correspondientes y se podrá agregar con la directiva *include* al fichero de configuración `automation: !include automations.yaml`

5.5 Scripts

Los scripts son una secuencia de acciones que Home Assistant ejecutará. Estas secuencias están disponibles como una entidad a través de un componente independiente, aunque también pueden incorporarse en una automatización.

La estructura básica de la sintaxis del script es una lista de mapas clave / valor que contienen acciones. Si una secuencia de comandos contiene solo una acción, la lista se puede omitir quitando los guiones. Se introducen al fichero de configuración mediante la sentencia script. El siguiente script de ejemplo de nombre example_script tiene dos servicios que son encender la luz de la identidad con id "light.ceiling" y enviar una notificación bajo el mensaje "Encendida la luz del techo".

```
script:
  example_script:
    sequence:
      - service: light.turn_on
        data:
          entity_id: light.ceiling
      - service: notify.notify
        data:
          message: 'Encendida la luz del techo'
```

5.6 Interfaz de usuario

Como hemos ido describiendo existen muchas configuraciones manuales que se pueden hacer desde la interfaz de usuario web. En este apartado se hará una descripción con más detalle de la interfaz de usuario.

Home Assistant cuenta con una aplicación nativa para iOS, mientras que para los clientes Android se puede configurar la página como pantalla de inicio en Chrome accediendo a la dirección http://ip-raspberry-pi:8123

La primera imagen que tenemos de la interfaz al iniciar HA es la que nos viene por defecto con dos sensores que son un símbolo meteorológico del sol y el usuario Santiago. Aquí se irán añadiendo los distintos sensores que se van añadiendo. En la zona de más abajo con la temperatura exterior es donde se irán añadiendo los interruptores, cámaras ,automatizaciones y scripts que se vayan configurando. Está en nuestra habilidad de organizar por grupos para dejar una interfaz impoluta y ordenada.

Figura 5-5. Panel inicial.

En la interfaz vemos en la esquina superior izquierda un cuadrito con tres rayas verticales, si clicamos en ellas podemos ver un menú desplegable con las siguientes opciones.

- **Resumen**: aquí aparecerá el cuadro de mandos domótico de todo lo que se configure.
- **Mapa**: obtenemos la ubicación de nuestro HA mediante la dirección IP, o porque se la hayamos indicado.
- **Registro**: muestra un registro secuencial donde se indica cómo van cambiando los estados de los dispositivos que hay integrados, indicando la hora y el orden en que se van sucediendo.
- **Historial**: este apartado hace un seguimiento de todo lo que sucede dentro de Home Assistant y le permite al usuario navegar a través de él.
- **Configuración**: como se ha ido describiendo a lo largo del punto 5, en este apartado se podrá realizar la configuración de los dispositivos y automatizaciones. Dentro se pueden ver diferentes apartados:
  - *Home Assistant Cloud:* es un servicio de paga para mantener el proyecto en la nube con ciertas ventajas en configuraciones.
  - *General*: su uso principal es validar el archivo de configuración.
  - *Customization*: personalización de los dispositivos, añadiendo nombre o iconos.
  - *Automation*: para generar las automatizaciones que queramos en nuestro hogar.

- *Script*: es igual que el apartado anterior, pero aquí sólo se definen los scripts o comandos que se ejecutan dentro de *automation*.

En el pie de página de este menú desplegable pordemos ver las herramientas de desarrollador "Developer tools" que constan de los siguientes apartados

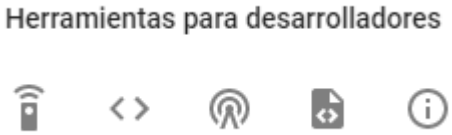


Figura 5-6. Herramienta desarrolladores.

Por order de izquierda a derecho en los iconos tenemos: servicios, estados, eventos, plantillas, información.

Las herramientas de desarrollo están diseñadas para que **todos** (no solo para los desarrolladores) prueben rápidamente cosas, como llamar a servicios, actualizar estados, generar eventos y publicar mensajes en mqtt ... etc.). También es una herramienta necesaria para aquellos que escriben automatizaciones personalizadas y scripts a mano.

6 MI SISTEMA DOMÓTICO

6.1 Instalación de Hassbian

Vamos a instalar Hassbian en nuestra Raspberry Pi, que como se comentó es una de las maneras más fáciles de instalación ya que es un sistema operativo que se basa en Raspbian (que a su vez se basa en Debian) y se distribuye con Home Assistant instalado de fábrica.

Comenzamos descargando la última versión de Hassbian accediendo a un repositorio de Github al que se podrá acceder por la página oficial de HA. En mi caso he descargado la última versión de Hassbian que es la 1.6.1

Una vez tenemos la imagen procedemos a grabarla en una tarjeta microSD de al menos 4 Gb. En mi caso he utilizado una de la marca Samsung EVO plus de clase 10 con 32 Gb. Para proceder al proceso de grabación es necesario la utilización de un software llamado Etcher, que también es posible utilizar otros como Win32DiskImager.

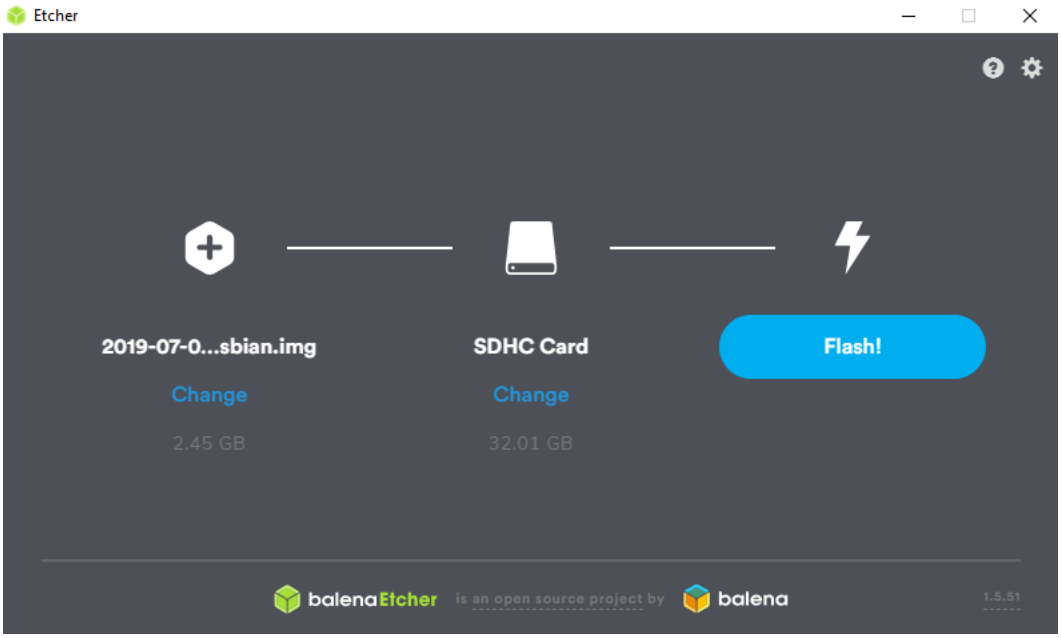


Figura 6-1. Etcher.

El siguiente paso es conectar la Rapsberry Pi mediante un cable ethernet a nuestro router, esto es fundamental antes de introducirle la tarjeta microsd y encenderla ya que la primera vez que se inicia, la Raspberry Pi se conectará a Internet y comenzará a descargar Home Assistant.

Es importante estar utilizando una fuente de alimentación adecuada. Los cargadores móviles pueden ser inadecuados, ya que algunos solo fueron diseñados para proporcionar suficiente energía al dispositivo para el que fue diseñado por el fabricante. No se debe intentar alimentar la Raspberry Pi desde el puerto USB de un televisor, computadora o similar.

Una vez tenemos en marcha Hassbian corriendo en nuestra Raspberry Pi, deberemos conocer la dirección IP que nuestro router le asigna que recibe para poder acceder mediante http://dirección-ip:8123/. Para ello accedemos a nuestro router vía web mediante la dirección 192.168.0.1 y en el aparecerá un listado de dispositivos que tenemos en casa con sus respectivas direcciones asignadas. Vemos que la dirección IP asignada a nuestro sistema es 192.168.0.163

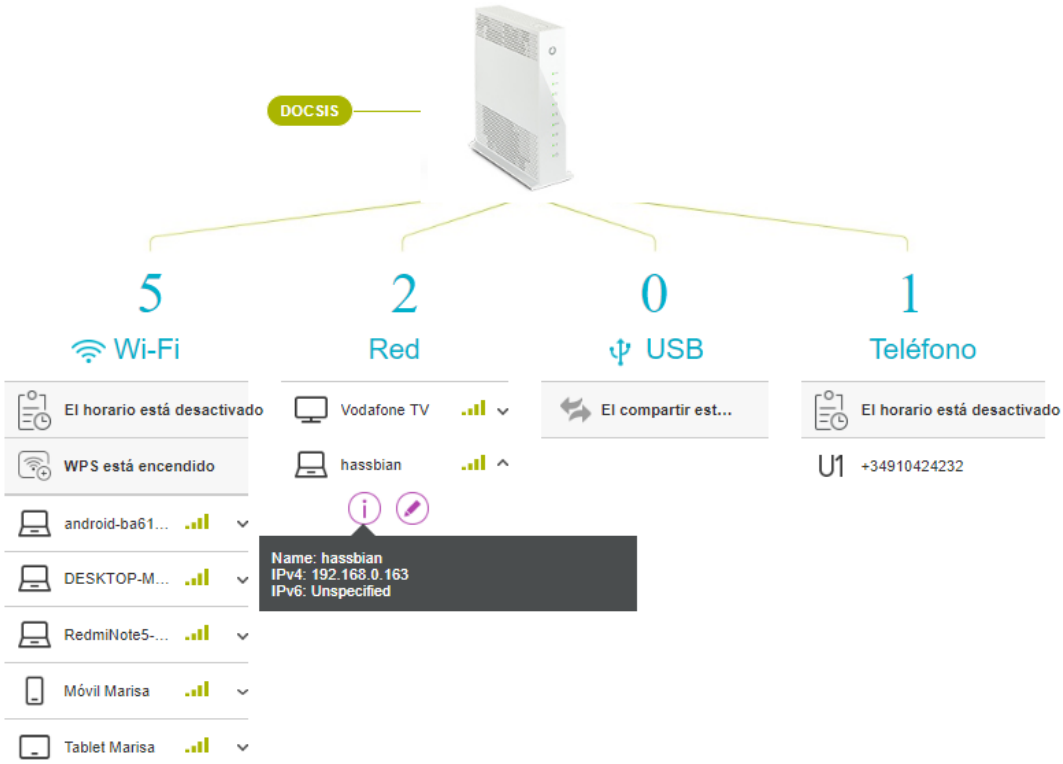


Figura 6-2. Mi router.

En el primer acceso a nuestro sistema nos pedirá crear una cuenta en la que tendremos que introducir nuestro nombre, nombre de usuario, y contraseña. La siguiente pestaña de configuración es para introducir el nombre que recibirá tu casa y la ubicación donde se encuentra. La zona horaria, y el sistema de unidades también es configurable en esta pestaña.

Una vez que se accede a http://192.168.0.163:8123/ y carga la página, significa que todo está configurado para la instalación básica. Home Assistant debería comenzar a detectar todos los dispositivos IOT de marca que pueda haber conectado a su red local. En este caso como no tengo ninguno activado mi pantalla de inicio sería la que aparece en la Figura 5-5.

6.2 Acceso SSH y configuración IP

Se utilizará el cliente *PuTTY* de Windows para acceder vía SSH a nuestro sistema con las credenciales iniciales por defecto que son ***pi*** y contraseña ***rapsberry.***

Abrimos el cliente y le proporcionamos la dirección IP asignada por el router y el puerto 22. Marcamos la opción SSH en el apartado *Connection type*. Pulsamos en *Open*.

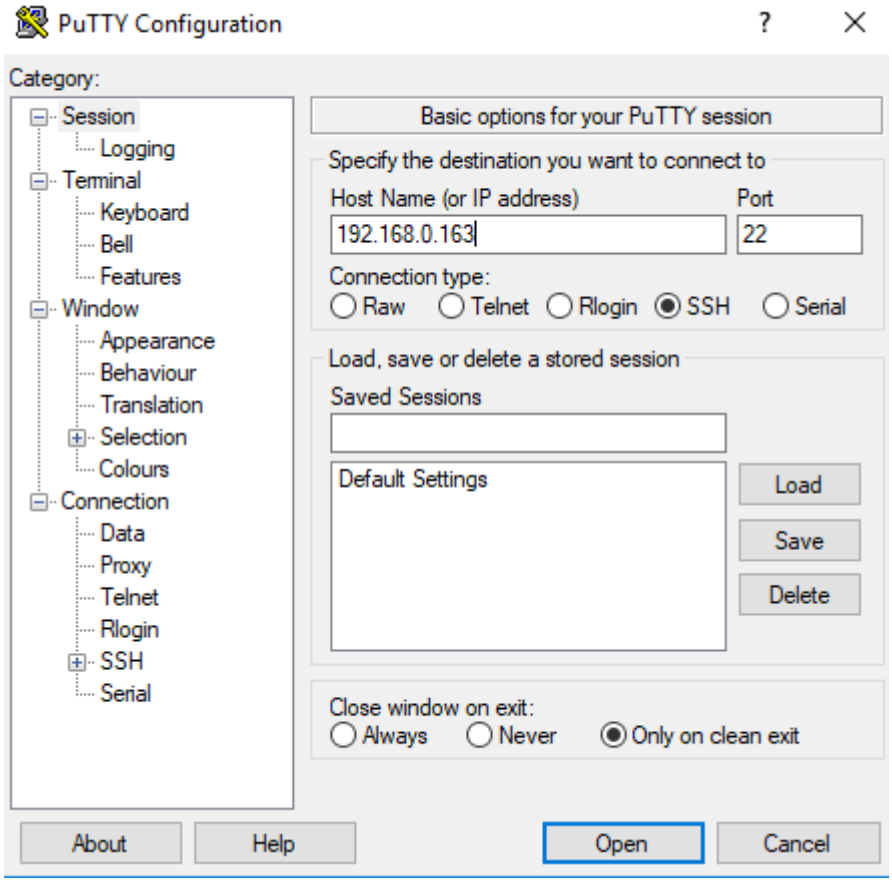


Figura 6-3. PuTTY.

Una vez introducidas las credenciales de acceso tendremos el terminal que nos indica que estamos dentro de nuestro sistema Hassbian y que podremos navegar a través de él mediante comandos. Habilidad más que adquirida en el grado.

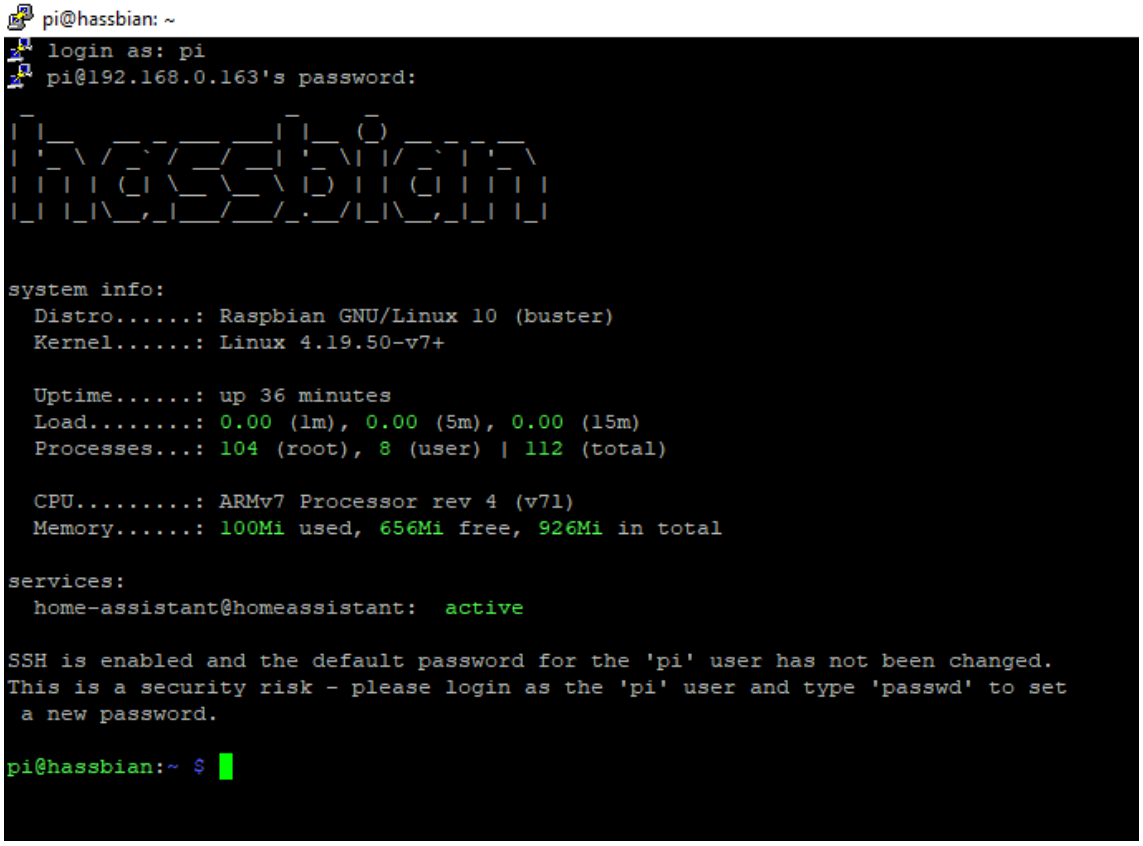


Figura 6-4. Terminal Hassbian.

Una vez dentro es necesario cambiar las credenciales de acceso ya que al ser las que se tienen configuradas por defecto haría nuestro sistema vulnerable.

```
sudo passwd pi
```

Nos pedirá el ingreso de una nueva contraseña de UNIX. Debemos recordar que esta contraseña distingue entre mayúsculas y minúsculas. Esta es la contraseña de acceso ROOT para su nuevo servidor. Es muy importante que recuerdar esta contraseña.

Con la ejecución del siguiente comando `sudo raspi-config` accederemos a una ventana en la que nos da diferentes opciones de configuración. Se procederá a cambiar la zona horaria en la 4 opción *Localisation options* eligiendo Europa y a continuación Madrid.

El siguiente paso es configurar la conexión Wi-Fi. No es un paso necesario ni obligatorio, pero ya que nuestro modelo de Raspberry Pi tiene integrada una tarjeta de conexión Wi-Fi haremos uso de ella y podremos ubicar nuestro sistema en el lugar que deseamos sin necesidad de atender a una conexión por cable.

Utilizamos el siguiente comando que abre el archivo *wpa_supllicant.conf* que contiene las credenciales de acceso a la red.

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Al final del fichero introducimos las credenciales de nuestro router

Network={

ssid="Robaphone"

*psk="*****"*

}

En *ssid* se introduce el nombre del punto de acceso (mi punto de acceso se llama "robaphone") y en *psk* la contraseña. Una vez introducido guardamos los cambios mediante la secuencia *CRTL + x, Y, Enter*. Por último, procedemos a introducir el comando `sudo reboot`.

Se nos sale del cliente puTTy y es debido a la nueva asignación de una dirección IP 192.168.0.156. El servidor DHCP (*Dinamic Host Control Protocol*) rara vez te asigna una dirección IP única y cada vez que nuestro sistema se reinicie y vuelva a acceder a la red puede ir variando su dirección IP por lo que sería muy engorroso tener que volver a buscarla y acceder de nuevo vía web.

Para esto se aporta la solución de asignarle a nuestro sistema una IP fija. Tenemos que decirle a nuestra Raspeberry Pi que no recurra al DHCP (que viene programado por defecto) y que en su lugar use siempre una dirección dada. La que queramos, pero siempre la misma, es decir, utilizar una IP estática.

Empezamos por conectaros vía SSH con nuestra nueva dirección 192.168.0.156. Una vez dentro vemos con ifconfig que esa es nuestra IP actual y vemos también nuestra tú mascara de red. Es necesario fijarnos en el apartado de *wlan0* ya que estamos conectados mediante la red inalámbrica.

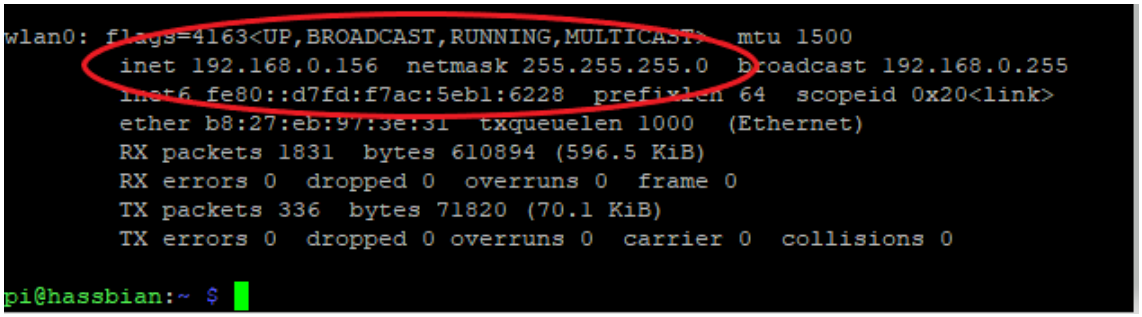


Figura 6-5. Averiguar IP

Podemos ver que efectivamente la dirección coincide y que la máscara de red es 255.255.255.0. Esto nos dice que nuestra direcciones IP varían en el rango de 192.168.0.* siendo * un número entre 0 y 255, aunque normalmente el 0 y el 255 suelen estar reservados. Usaré 200 como dirección, por lo que nuestra IP quedará 192.168.0.200

Con el comando route –ne vemos el router por defecto de mi red y sabemos que tiene la dirección gateway 192.168.0.1.

Ahora tenemos que irnos al fichero de nuestro sistemas que queremos editar y accedemos mediante el comando sudo nano /etc/dhcpcd.conf. Añadimos las siguientes líneas.

#configuracion ip estatica

interface wlan0

static ip_address=192.168.0.200/24

static routers=192.168.0.1

static domain_name_servers=8.8.8.8 8.8.4.4

Es necesario acceder a la configuración del router para deshabilitar que el servidor DCHP asigna la dirección usada por nuestro sistema como estática y así evitar problemas

6.3 Samba

Hasta este instante cada vez que debemos configurar algo de nuestro sistema, es decir, archivos en Hassbian debemos acceder vía SSH y movernos en el terminal mediante comandos. Para acabar con esto podemos utilizar el protocolo Samba.

Samba es una implementación de código abierto del protocolo Server Message Block (SMB). Los sistemas operativos Microsoft Windows y OS/2 utilizan SMB para compartir por red archivos e impresoras y para realizar tareas asociadas. Gracias al soporte de este protocolo, Samba permite a las máquinas Unix entrar en el juego, comunicándose con el mismo protocolo de red que Microsoft Windows y aparecer como otro sistema Windows en la red (desde la perspectiva de un cliente Windows). El servidor Samba ofrece los siguientes servicios:

- Compartir uno o varios sistemas de archivos
- Compartir uno o varios sistemas de archivos distribuidos
- Compartir impresoras instaladas en el servidor entre los clientes Windows de la red
- Ayudar a los clientes permitiéndoles navegar por la red
- Autentificar a los clientes que ingresan en un dominio Windows
- Proveer o ayudar con un servidor de resolución de nombres Windows (WINS).

En resumidas cuentas, es un modo más fácil de acceder a los archivos de Raspberry (Hassbian) mediante nuestro PC que trabaja con Windows.

Solo existe un paso a seguir y es la ejecución del comando `sudo hassbian-config install samba`.

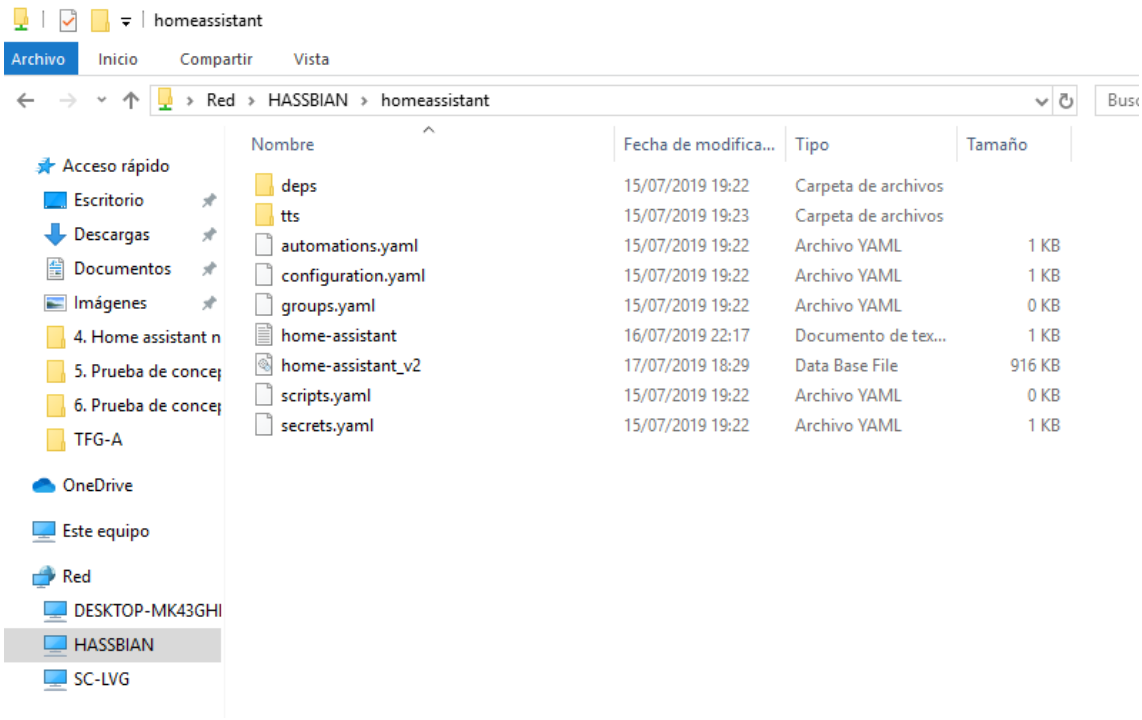


Figura 6-6. Samba.

6.4 Acceso remoto

Cuando accedemos Internet, generalmente lo hacemos a través de nuestra dirección IP pública, una IP que además suele ser dinámica (es decir, que cambia cuando reiniciamos el router o cada cierto tiempo). Lo que es inviable si por ejemplo queremos mantener un servidor conectado a Internet o queremos tener disponible nuestro ordenador, o un dispositivo de nuestra red local para conectarnos a él desde cualquier lugar. Si se produce un reinicio del mismo podríamos obtener una dirección IP diferente y no podríamos acceder a nuestro sistema. Para ayudarnos a solucionar estos inconvenientes existen los DDNS, o servidores DNS dinámicos, como Duck DNS. Por lo que servicios gratuitos como Duck DNS nos permite tener localizados nuestros dispositivos en internet como si tuvieran una IP fija.

Un DDNS es un servicio (gratuito o de pago) diseñado para convertir nuestra dirección IP pública, complicada de recordar, en un dominio mucho más sencillo de recordar y, además, mantenerlo siempre actualizado de manera que podamos estar seguros de que la conexión con nuestra red está siempre garantizada, incluso aunque tengamos IP dinámica.

Existen muchos servidores DDNS disponibles en la red. Algunos de ellos son de pago, como DynDNS**,** y otros gratuitos, como No-IP, pero muy molestos al estar cada poco tiempo pidiendo pagar por el servicio o dando de baja nuestro DDNS si no lo renovamos. Duck DNS es un servidor DNS totalmente gratuito que nos permite hacer uso de este tipo de servicios sin tener que pagar por él y, sobre todo, sin molestos avisos cada poco tiempo para renovarlo manualmente.

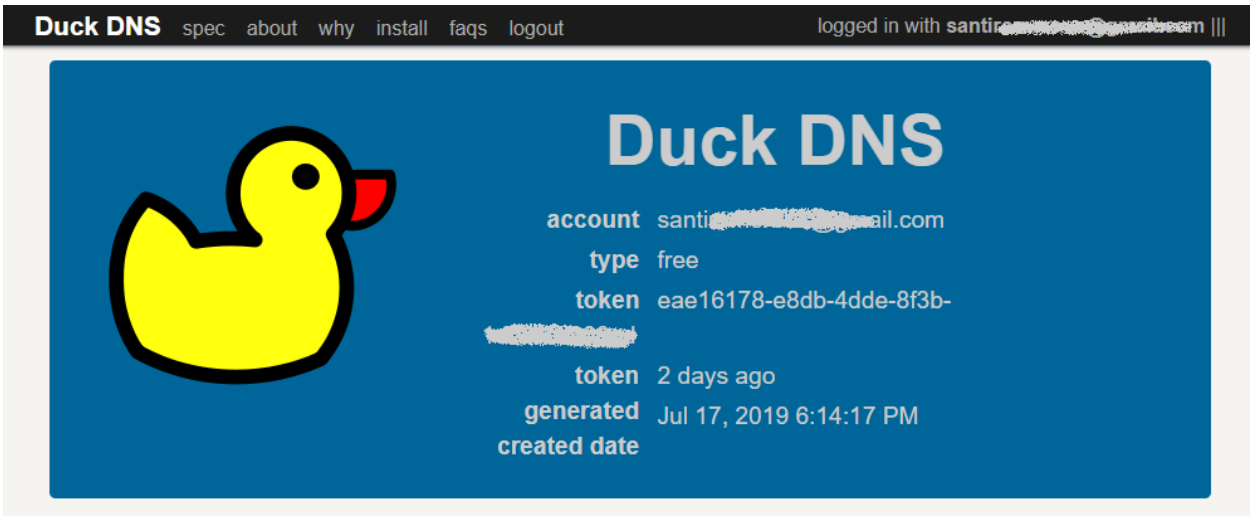


Figura 6-7. Duck DNS.

Por lo tanto para usar nuestro sistema desde internet fuera de nuestra red local se implementará Duck DNS + LetsEncrypt para tener un servicio DN gratuito y con un tráfico seguro encriptado.

Lo primero es acceder a la pagina de DuckDNS https://www.duckdns.org/ registranos de alguna de las formas posibles, en mi caso elijo por cuenta de Gmail. Una vez registrado crearemos un domino que será con el que podamos acceder desde internet. Mi dominio será ***santihome.duckdns.org.*** También se nos asigna una IP publica y un token que usaremos mas adelante.

El siguiente paso es dirigirse a la pestaña *Install* de la web y seguir los pasos marcados.

```
ps -ef | grep cr[o]n
mkdir duckdns
cd duckdns
nano duck.sh
```

Se introduce nuestro token e id en el fichero duck.sh mediante esta sentencia *echo url="https://www.duckdns.org/update?domains=santihome&token=eae16178-e8db-4dde-8f3*-**********&ip=" | curl -k -o ~/duckdns/duck.log -K -*

Los astericos están puestos por motivos de seguridad y no enseñar mi token. Guardamos el fichero y ejecutamos los siguientes comandos.

```
chmod 700 duck.sh
crontab -e
```

Introducimos la siguiente línea en el fichero que se nos abre **/5 * * * * ~/duckdns/duck.sh >/dev/null 2>&1* que hace que duck.sh se ejecute cada 5 min

Debemos entrar en nuestro router y abrir el puerto que queramos y redigirlo internamente a la ip de nuestra raspberry (192.168.0.200) previamente definida como estática. La configuración en nuestro router deberá quedar como la siguiente imagen.

Figura 6-8. Redirección de puertos.

Una vez llegados a este punto ya es posible acceder mediante la dirección http://santihome.duckdns.org:8123.

6.5 Detección de dispositivos

En este apartado se explicará cómo hacer un seguimiento de los dispositivos que tenemos conectados a la red en casa. Es de especial interés cuando en casa viven familias, para poder así tener un control de quién está en casa.

Par ello se utilizará Nmap (Network Mapper) que es un software libre para explorer, administrar y auditor la seguridad de redes de ordenadores. Detecta hosts online, sus puertos abiertos, servicios y aoplicaciones corriendo en ellos.

Lo primero es abrir una conxexión SSH con nuesro Sistema pra poder accede al terminal. Una vez dentro instalamos el software mediante el siguiente comando

```
sudo apt-get install net-tools nmap
```

Una vez concluida la instalación dentro de nuestro fichero de configuración añadiremos las siguientes sentencias

```
device_tracker:
platform: nmap_tracker
          host:192.168.0.1/24
```

Una vez realizado el cambio guardamos y restauramos HA. Si entramos en la interfaz de usuario vemos que ha detectado todos los dispositivos conectados por wifi en casa.

Los dispositivos que encuentra los almacena en el fichero *known_hosts.yaml* el cual vamos a modificar para que solo nos aparezcan los dispositivos que nosotros como usuarios queramos. En mi sistema solo se hará seguimiento a tres dispositivos y son los tres móviles de los habitantes del hogar (Emilio, Marisa, Santi).

Abrimos el fichero *known_hosts.yaml* y vemos que por cada dispositivo aparecen la siguientes líneas

```
80_35_c1_60_40_58:
  hide_if_away: false
  icon:
  mac: 80:35:C1:60:40:58
  name: Santi
  picture: /local/imagensanti.jfif
  track: true
```

En `name` se introduce el nombre del usuario que aparecerá en a interfaz de usuario. En `picture` se introduce una imagen que también aparecerá. Y por último y más importante la propiedad `track` si el valor es true hará un seguimiento, por lo que se debe poner a false a todos los dispositivos que no nos importan.

Por último, se le añadirán las siguientes líneas al componente `device_tracker`

```
device_tracker:
    - platform: nmap_tracker
      hosts: 192.168.0.1/24
      interval_seconds: 10
      consider_home: 180
      scan_options: " -privileged -sP -host-timeout 10s "
      new_device_defaults:
        track_new_devices: False
        hide_if_away: True
```

Estás sentencias hacen que cada diez segundos se haga un nuevo rastreo de los dispositivos en casa y que además se esperen 180 segundos para hacer que cuando un dispositivo no esté en casa pase al estado de ‘not_home’. Esto se hace debido a que los terminales tienen configurados ahorro de energía en su configuración Wi-Fi.

Una vez que tenemos esto configurado, podemos jugar con la presencia de estos dispositivos, ya que, estarán con el estado “home” cuando estén detectados o “not_home” cuando se encuentran fuera de la LAN, por lo que podemos usarlos para o bien lanzar algún *trigger* en una automatización (por ejemplo, el dispositivo entra o se va) o, hacer que sean la condición para que se ejecute algo, por ejemplo, si está o no en casa.

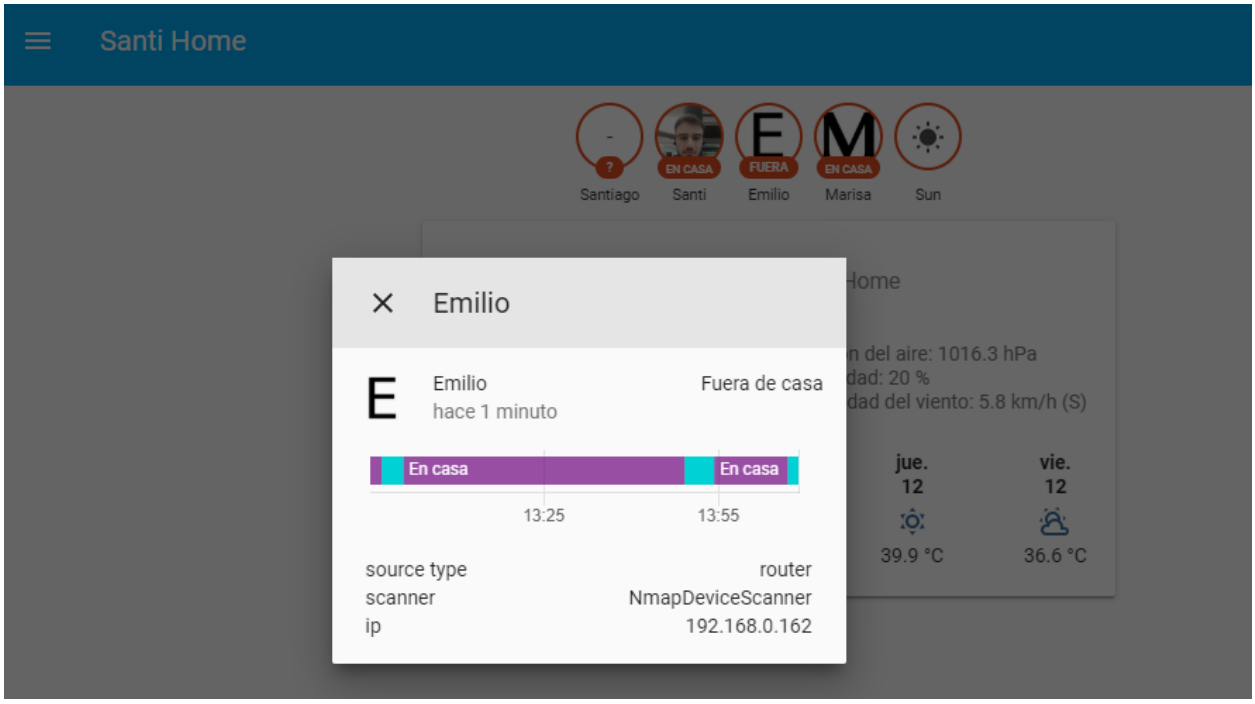


Figura 6-9. Rastreo de dispositivos.

6.6 Sensor DHT22

El sensor DHT22 es un sensor de bajo costo que mide la temperatura y humedad del relativa en el lugar donde se coloca. La temperatura la mide en el rango de los -40ºC a los 80ºC con una precisión de 0.5ºC y del 0% al 100% la humedad relativa con una precisión del 2%.

El sensor se conectará al GPIO de la Raspberry Pi mediante tres cables:

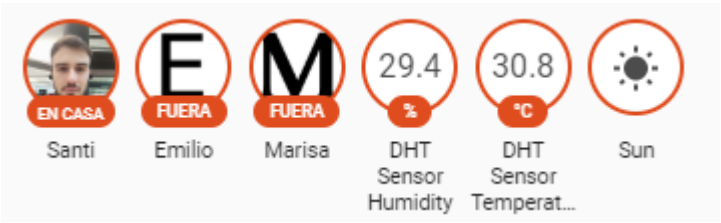
- Uno gris para el positivo que se conecta al pin 2 que nos da los 5V.
- Uno púrpura para la salida de datos que se conectará en el pin 7.
- Uno azul para el negativo que se conectará a tierra en el pin 6.

Con este sensor mediremos la temperatura y humedad del salón que es donde tenemos nuestro hub como centro de control. Una vea se ha colocado correctamente nos dirigimos hacia el archivo de configuración para añadir lo siguiente.

```
sensor:
  #Sensor dht22 temperatura y humedad del salon
  - platform: dht
    sensor: DHT22
    pin: 4
    monitored_conditions:
      - temperature
      - humidity
```

En el pin se introduce cuatro ya que el siete mencionando arriba se corresponde con el GPIO 4.

Una vez realizados estos cambios se guarda el fichero de configuración, se reinicia HA y veremos que al recargar ya aparecen los dos sensores en nuestro panel.



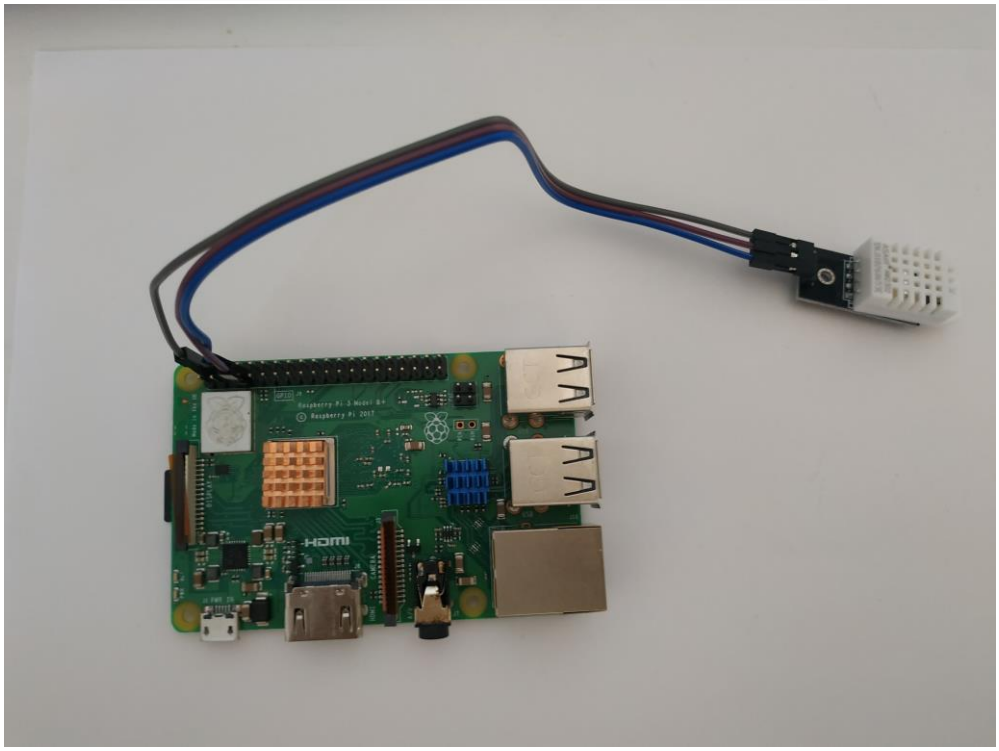


Figura 6-10. Sensor DHT22.

Si pinchamos en el icono de los sensores se podrá ver un histórico de las dos magnitudes.

6.7 Enchufe Smart Life

En este apartado se va a instalar un enchufe inteligente con el que podemos controlar el paso de la corriente y poder tener el control sobre su encendido o apagado. Este enchufe será colocado en una toma de mi salón. Con esta solución se podrá tener un control de su encendido y apagado.

Figura 6-11. Enchufe.

Lo primero es descargarse la aplicación en nuestra *App Store* que recibe el nombre de *Smart Life.* Una vez dentro procederemos a crear una cuenta de usuario. Una vez nos hayamos registrado, accedemos al panel con el símbolo + , añadimos nuestro dispositivo de toma eléctrica y configuramos el acceso a nuestra red Wi-Fi.

Ahora nos dirigimos al fichero de configuración, y es necesario

recordar el usuario y contraseña con el que nos dimos de alta porque será necesario al incluir las siguientes líneas

```
tuya:
   username: correodealta@gmail.com
   password: lacontraseñaquesepuso
   country_code: 34
   platform: smart_life
```

Para una mayor seguridad en el uso de contraseñas, es mejor opción la utilización del archivo *secrets.yaml* dentro del cual se introducirá la contraseña y en nuestro archivo de configuración se referenciará.

Dentro del archivo *secrets.yaml* se introduce la siguiente sentencia

```
enchufe_pass: lacontraseñaquepusimos
```

Y dentro del archivo de configuración se cambiará lo siguiente

```
password: !secret smartlifpass
```

Recordar siempre cambiar al nombre a la entidad para hacer un uso de ella más amigable.

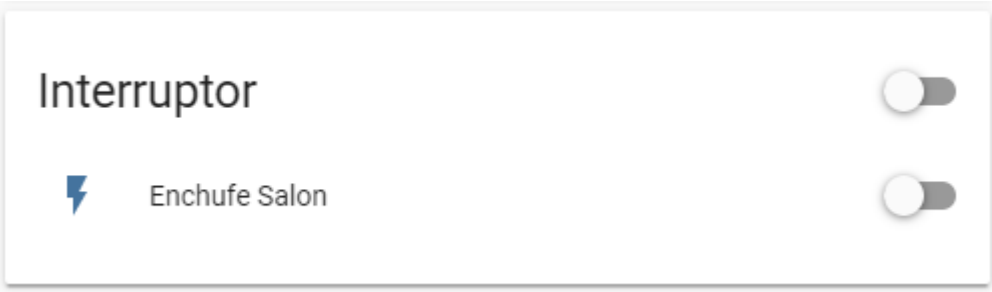


Figura 6-12. Enchufe en HA.

6.8 Bombillas

6.8.1 Bombilla Yeelight

En este apartado se marcarán las pautas de instalación de la bombilla de la marca Xiaomi Yeelight. Esta bombilla tiene conexión Wi-Fi, realiza un consume de 8 vatios y ofrece más de 16 millones de colores. Además, cuenta con un certificado de eficiencia energética A+, lo que supone que realiza un bajo consume eléctrico. Es posible ajustar el brillo y la temperatura de color.

Lo primero que debemos hacer es colocar nuestra bombilla sin hacer ningún tipo de configuración aún y probar que funciona correctamente de manera común, es decir mediante el interruptor. El Segundo paso es descargar la aplicación de Yeelight desde *Play Store* (en mi caso soy usuario de Android), instalarla y abrirla. Un vez abierta pulsamos en *Experimentar Ahora,* dejamos seleccionado el servidor que nos viene como recomendado, y pulsamos sobre *Iniciar Sesión.* Si no tenemos cuenta, deberemos crearnos una mediante la misma aplicación. Con la bombilla encendida, en la aplicación pulsamos sobre *Añadir Dispositivo* y seguidamente sobre *Bombilla LED (color).*

Pulsamos sobre *Siguiente,* se produce un escaneo de la red y encuentra nuestro dispositivo referenciado mediante una dirección física (MAC). Pulsamos sobre la dirección y *Siguiente.* Por ultimo nos pedirá acceso a la red y le deberemos dar las credenciales de nuestra Wi-Fi. Una vez terminados estos pasos, dentro de la aplicación seleccionamos la bombilla, mostramos todas las opciones, pulsamos sobre *Control LAN* y habilitamos esta

opción.

Una vez que tenemos la instalación de la bombilla realizada accedemos a HA y lo reiniciamos. Vemos que nos aparece directamente la luz en el panel ¿pero por qué? Esto es debido a que tenemos el componente *discovery* en nuestro fichero de configuración por lo que se ha configurado de manera automática. Si no tuviéramos este componente bastaría con poner las siguientes líneas.

```
light:
   -platform: yeelight
   devices:
      192.168.0.169:
      name: Luz Habitación
```

Figura 6-13. Bombilla Yeelight.

Una vez que todo está instalado podemos ver en la interfaz gráfica que la bombilla nos aparece con un nombre poco amigable por lo que sería recomendable cambiarlo y darle un nombre que nos ayude a identificar donde está ubicada esa entidad, por ejemplo. Para ello nos dirigimos a la pestaña *Configuración* dentro de ella pinchamos sobre *Personalización* y tendremos un campo *Name* en el que podemos cambiar el nombre a por ejemplo "Luz Habitación".

Otra manera de realizar este cambio es añadiendo en el fichero de configuración `customize` dentro de `home assistant`. Para una mejor legibilidad se creará un archivo llamado *customize.yaml* donde se podrán añadir todas las modificaciones que se quiera y este se incluirá dentro del archivo de configuración

```
homeassistant:
    name: santi home
    customize: !include customize.yaml
```

Dentro del archivo *customize.yaml* tenemos lo siguiente

```
light.yeelight_color1_04cf8ca2deda:
    friendly_name: Luz Habitación
```

6.8.2 Bombilla Smart life

Esta bombilla tiene unas características parecidas a la Yeelight, con capapicades de ajuste del brillo y cambios de color. Para instalarla es necesario tener la aplicación Smar Life, y aprovechando que ya tenemos una cuenta en activo solo debemos hacer un escaneo en red con la aplicación para detectar la bombilla.

No hace falta añadir más código al archivo de configuración ya que con el descrito en el punto 6.7 la integración funcionará correctamente ya que estamos usando la misma plataforma que el enchufe. Se cambiará el nombre de la bombilla a uno más amigable como “Luz salón” utilizando `friendly_name` en el *customize.yaml* o por la interfaz gráfica (que es lo mismo).

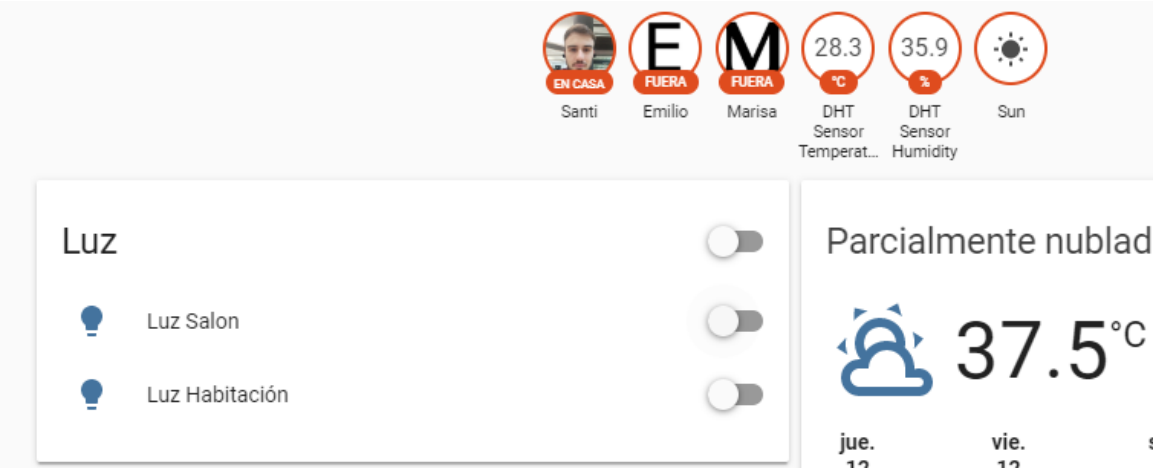


Figura 6-14. Bombillas HA.

6.9 Sonoff Touch

Sonoff Touch es un interruptor de pared para controlar el encendido y apagado de alguna de nuestras luces. Es un interruptor inteligente al cual se puede acceder mediante nuestra red Wi-Fi y poder ejercer un control inalámbrico sobre él. Cabe destacar que para accionarlo manualmente no dispone de un accionando mecánico si no que es táctil.

Figura 6-16. Delantera Sonoff.

Figura 6-15. Delantera Sonoff.

Para instalar el interruptor es necesario primero reemplazar el interruptor de la zona que queremos controlar. En este caso quiero ejercer un control sobre las luces de mi salón.

En primer lugar es necesario cortar la corriente eléctrica ya que vamos a manipular cables, una vez realizado esto y habiendo reemplazado nuestro enchufe, procedemos a colocar los cables de tierra, fase y neutro en nuestro conexionado tal y como se puede observar en la Figura 6-16. Una vez colocado nuestro interruptor deberemos comprobar que funciona sin ningún problema.

Descargaremos la aplicación en nuestra *App Store* llamada *eWelink* y en ellas procederemos con los pasos marcados en otros apartados, registro, inicio de sesión, descubrimiento del dispositivo en nuestra red, etc. Una vez terminados estos pasos procederemos a la introducción del dispositivo en HA.

HA no tiene de forma nativa el control sobre dispositivos Sonoff por lo que habrá que instalar componentes personalizados. En este enlace https://github.com/peterbuga/HASS-sonoff-ewelink hacia un repositorio de GitHub podremos encontrar un componente personalizado para Home Assistant que nos permitirá utlizar los dispositivos de Sonoff. Una vez lo tenemos descargado y descomprimido deberemos copiar el directorio *sonoff* dentro de nuestro directorio */homeassistant/custom_components.* El directorio *sonoff* contiene tres archivos python con el que se podrá inicializar el componente sonoff sin necesidad de cambiar el firmware original, ya que está opción es algo más compleja.

En HA es necesario añadir el en el fichero de configuración el siguiente código

```
sonoff:
     username: santiromerous@gmail.com
     password: !secret sonoffpass
     scan_interval: 10
     grace_period: 600
     api_region: 'eu'
```

Una vez realizado todos estos pasos, es de interés comentar el problema al que me enfrenté en la isntalación de este este interruptor, y es que al seguir todos los pasos y reiniciar HA me aparecía un error *"Invalid config, sonoff could be not set up"*. Tras buscar en el foro de GitHub donde se exponen los problemas que se tienen y depurando el error mediante la consola de HA que te permite seguir los errores que se producen en el sistema, encontré la solución. El error marcado era el siguiente.

```
Error loading custom_components.sonoff. Make sure all dependencies are installed
Traceback (most recent call last):
  File "/srv/homeassistant/lib/python3.7/site-packages/homeassistant/loader.py", line 263, in
_load_file
    module = importlib.import_module(path)
  File "/usr/local/lib/python3.7/importlib/__init__.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 1006, in _gcd_import
  File "<frozen importlib._bootstrap>", line 983, in _find_and_load
  File "<frozen importlib._bootstrap>", line 967, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 677, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 728, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "/home/homeassistant/.homeassistant/custom_components/sonoff/__init__.py", line 28, in
<module>
    import websocket
ModuleNotFoundError: No module named 'websocket'
```

HA es instalado en un entorno virtual dentro de nuestro sistema Rapsbian basado en Linux, y el problema es que el archivo de inicialización del componente *__init__py* no podía importar el modulo *'websocket'*. Por lo que seguí los siguientes pasos.

1. sudo –u homeassistant –H –s (para cambiar al usuario de HA)
2. cd /srv/homeassistant/bin (para moverme hacia el entorno virtual)
3. . activate (activacion del entorno virtual)
4. pip install websocket-client

Una vez instalado el interruptor sin ningún problema se procederá a cambiar el nombre que aparece en la interfaz gráfica a "Luz cocina".

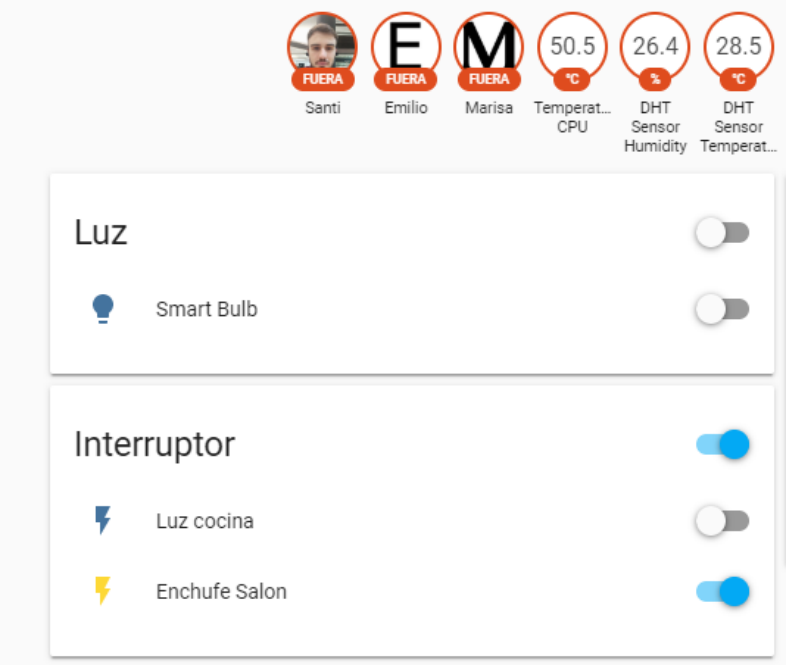


Figura 6-17. Luz cocina.

6.10 Notificaciones telegram

Bajo mi punto de vista una de las cosas más importante a la hora de dotar a tu casa de inteligencia es el hecho de que el sistema tenga la capacidad de hablar con nosotros, es decir, de ser capaz de notificarnos y sobre todo que estas notificaciones nos vengan dadas en una aplicación la cual usemos día a día para una lectura más fácil.

Existen multitudes formas de notificación, pero la elegida en este proyecto es Telegram, un sistema de mensajería instantánea para *smartphones* parecida a WhatsApp pero que te permite crear bots con los que podremos interactuar. Por lo que el primer paso será crear un bot en Telegram.

Pasos a seguir:

1. Buscamos el usuario BotFather, que es, como el nombre indica, el padre de todos los bot. Se usa para crear nuevas cuentas de bot y gestionar las existentes
2. Introducimos por teclado */start* y aparecerán una lista de comandos que se podrán usar.
3. Introducimos el comando */newbot* e introducimos el nombre que queramos. En mi caso el nombre introducido es SantiHome
4. En este paso es necesario introducir el nombre de usuario acabado de alguna manera en bot, por ejemplo, SantiHome_bot
5. Ya tenemos creado el bot y Botfather nos da una clave para poder hacer uso del bot por web.

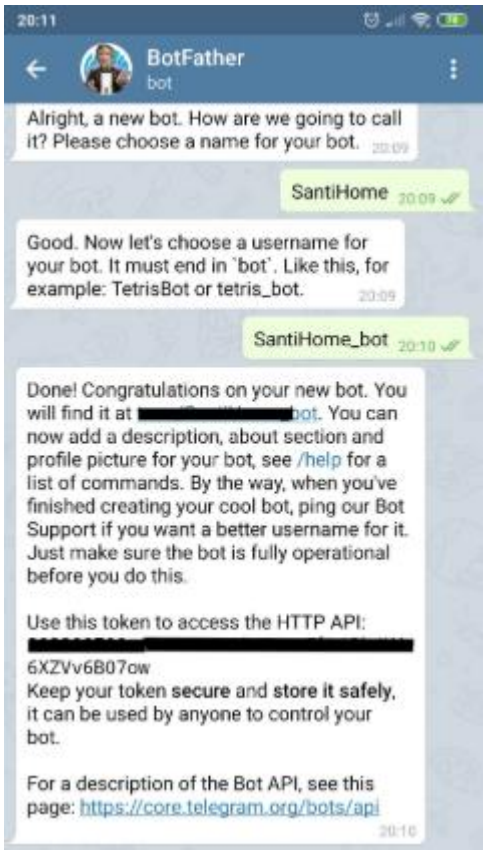


Figura 6-18. BotFhater.

Para poder integrar Telegram en mi sistema es necesario tan solo un dato, el ID de conversación para que nuestro bot sepa a quien hablar, para eso, buscamos nuestro bot en la aplicación mediante su nombre. Escribiremos el comando */start* y posteriormente el comando */getid* y contestará con un número. Con ese número apuntado, ya podemos configurar nuestro Home Assistant para que pueda notificarnos mediante Telegram.

Home Assistant permite diferentes maneras de usar Telegram ya que, el propio Telegram permite varias maneras de interactuar con los bots, la opción utilizada es "Telegram Polling" para poder mandar mensajes a nuestro usuario en Telegram. Para ello, abriremos la configuración y añadiremos lo siguiente:

```
telegram_bot:
   - platform: polling
     api_key: AAAA1234
     allowed_chat_ids:
       - BBBB1234

notify:
   - name: telegramsanti
     platform: telegram
     chat_id: BBBB1234
```

En AAA1234 se introduce la clave que se nos suministró a la hora de crear el bot y en BBB1234 se introduce el ID de conversación.

Es posible la comunicación con más usuarios introduciendo dentro de `allowed_chat_ids` y `chat_id` los diferentes ID de conversaciones que queramos añadir.

Para probar que la integración de la notificación funciona he creado una automatización que me avisa por

telegram cuando la luz de la habitación se enciende.

```
automation:
   - alias: 'Notifica luz habitacion ON'
     trigger:
       - platform: state
         entity_id: light.yeelight_color1_04cf8ca2deda
         from: 'off'
         to: 'on'
     condition: []
     action:
       service: notify.telegramsanti
       data:
       message: "Luz habitacion encendida"
```

Por lo que hago la comprobación y enciendo la luz desde HA y efectivamente se me notifica mediante Telegram el mensaje “Luz habitación encendida”

Figura 6-19. Automatización Luz.

6.11 Información sobre el Sistema

6.11.1 Información sobre Raspberry Pi

En este apartado se detallará como llevar a cabo un control sobre los parámetros de nuestra Rapsberry Pi, parámetros tales como uso de la memoria, tarjeta microSD, temperatura, … entre otros. Para ellos se utilizarán dos plataformas que no darán la información, *systemmonitor* y *command_line.*

La plataforma *systemmonitor* (se utiliza como un `sensor`) permite monitorear el uso del disco, el uso de la memoria, el uso de la CPU y los procesos en ejecución.

```
- platform: systemmonitor
  resources:
    - type: disk_use_percent
      arg: /
    - type: memory_use_percent
    - type: memory_free
    - type: processor_use
    - type: last_boot
```

La plataforma *command_line* se puede utilizar para emitir comandos específicos para obtener datos. Esta podría convertirse en nuestra plataforma más poderosa, ya que permite a cualquier persona integrar cualquier tipo de sensor en Home Assistant que pueda obtener datos desde la línea de comandos.

```
- platform: command_line
  name: Temperatura CPU
  command: "cat /sys/class/thermal/thermal_zone0/temp"
  unit_of_measurement: "°C"
  value_template: '{{ value | multiply(0.001) | round(1) }}'
```

Veremos como en la pantalla principal de la interfaz de usuario nos aparecen en la parte de los sensores todos los que acabamos de añadir relativos al estado del sistema.

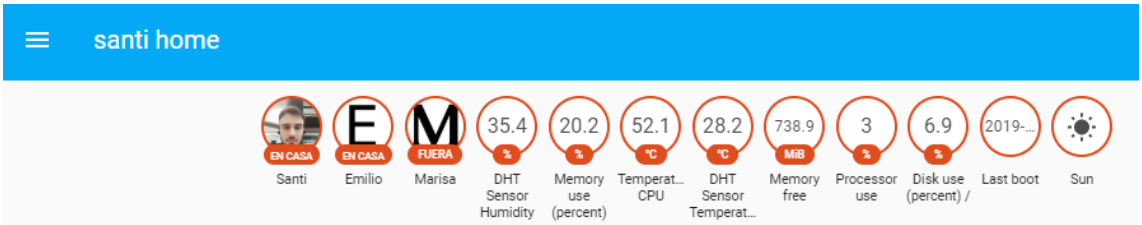


Figura 6-20. Parámetros del sistema.

El más importante es el de la temperatura ya que nos indica los grados a la que está la CPU, parámetro muy importante para que podamos llevar el control y no sufra daños. La temperatura no debería subir más de 70ºC ya que la Raspberry cuenta con disipadores de calor, por lo que se creará una automatización que nos enviará una notificación al móvil si la temperatura pasa de los 65ºC, para poder actuar en consecuencia.

```
- id: 'Notificacion temperatura cpu'
  alias: Notificacion temperatura CPU
  trigger:
  - above: '65'
    entity_id: sensor.temperatura_cpu
    platform: numeric_state
```

```
    condition: []
    action:
    - data:
        message: La temperatura de la CPU ha llegado a los {{
states.sensor.sensor.temperatura_cpu.state
          }} °C
      service: notify.telegramsanti
```

6.11.2 Información sobre HA

También es importante conocer información acerca de Home Assistant para poder sacar el mejor rendimiento posible a nuestro sistema. La plataforma *version* dentro de sensor nos da información acerca de las versiones actuales de HA.

```
Sensor:
   - platform: version
     name: Version instalada HA
```

Existen otras muchas formas de conocer la versión de la plataforma, por ejemplo, usando *command_line*

```
  sensor:
   - platform: command_line
     name: Version HA
     command: "/home/homeassistant/bin/hass --version"
```

Otra plataforma interesante de comentar es *scrape* que mediante ella podemos recopilar información de sitios webs, cargando una página HTML con la opción de buscar valores. Solo funciona con páginas webs simples.

La última versión de Home Assistant es publicada en https://www.home-assistant.io/, por lo que añadiendo el siguiente código en nuestro archivo de configuración tendríamos siempre información sobre la última versión disponible.

```
    - platform: scrape
      resource: https://home assistant.io
      name: Última versión disponible
      select: ".current version h1"
      value_template: '{{ value.split(":")[1] }}'
```

Y además con las siguientes líneas añadiré un sensor que me dirá si existe una versión nueva disponible con un "Si" o un "No" comparando los dos sensores añadidos, tanto el de última versión disponible como el de versión HA.

```
    - platform: template
      sensors:
          ha_version_update:
            friendly_name: '¿Nueva versión?'
            value_template: >-
              {%   if   states('sensor.Ultima_version_disponible')   ==
states('sensor.Version_instalada_HA') %}
                Sí
              {% else %}
                No
              {% endif %}
```

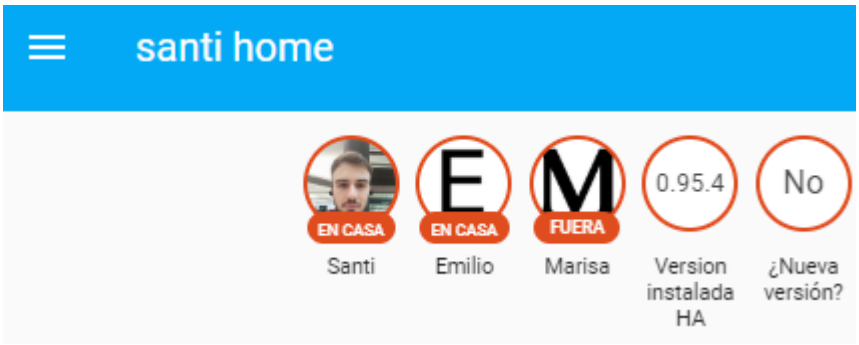


Figura 6-21. Información HA.

6.12 Control de precio en Gearbest

Home Assistant nos permite la integración de un sensor que controla el precio de un producto de Gearbest para poder realizar con la variación de precios lo que queramos, de esta forma, si estamos esperando a que se ponga de oferta, podemos hacer que no nos notifique.

Gearbest es una de las empresas más grandes de venta online de productos de todo tipo, aunque, principalmente tecnológicos, la cual personalmente utilizo en muchas de mis compras. La utilidad de esto es enorme ya que nos ahorra tiempo a la hora de ver si el producto baja de precio.

Dentro de nuestro *configuration.yaml* insertaremos lo siguiente

```
  - platform: gearbest
    currency: EUR
    items:
     -  url: se introudce la URL del producto a comprar
        name: miband4
```

Y además creo una automatización que me avisa a telegram para cuando el pecio del prodcuto baje de los 25 €

```
- id: 'precio gearbest'
  alias: preciogearbest
  trigger:
  - below: '65'
    entity_id: sensor.miband4
    platform: numeric_state
  condition: []
  action:
  - data:
      message: El precio de la miband4 ha bajado de 25€
    service: notify.telegramsanti
```

6.13 Tiempos de recorrido

En HA es possible crear un sensor que nos informe sobre el tiempo de viaje de un punto a otro, una gran utlidad ya que también tiene en cuenta el tráfico. Para ello utiliza la tecnología "Waze", una aplicación social de tránsito automotor en tiempo real y navegación asistida por GPS.

Para hacer uso de este sensor en HA es necesario en primera instancia crear una archivo llamado *zonas.yaml* e incluirlo dentro de nuestro archivo de configuración mediante `zone: !include zonas.yaml`. La estructura del archivo es sencilla, asignar un nombre del lugar donde queremos marcar el tiempo, y dar sus coordenadas. A continuación, se muestra su estructura mediante un ejemplo, pero el resto del fichero se encuentra en el Anexo A.

```
- name: Trabajo
  latitude: 40.407389
  longitude: -3.788592
  radius: 120
  icon: mdi:worker
```

Se tienen dos opciones para averiguar el tiempo del viaje, podemos hacer un sensor por cada ruta o crear una lista con posibles destinos.

- Un sensor por ruta

```
sensor:
  - platform: waze_travel_time
    name: "Tiempo al Trabajo"
    origin: zone.home (zona configurada al inicio del fichero de configuración)
    destination: zone.trabajo (La zona destino que queramos, en este caso es trabajo)
    region: 'EU'
```

- Una lista con posibles destinos

Par crear un sensor donde podamos elegir diferentes orígenes y destinos para saber los tiempos, primero se debe crear un *input_select* con las opciones que queramos. Además, podemos mezclar ambas con origen y destinos.

```
input_select:
  destino:
    name: Destino
    options:
      - Casa
      - Trabajo
      - Casa Jose Luis
      - Trabajo Pabellon entrenamiento
      - Local ensayo
  origen:
    name: Origen
    options:
      - Casa
      - Trabajo
      - Casa Jose Luis
      - Pabellon entrenamiento
      - Local ensayo
```

A continuación se crea un sensor template con los datos del origen y destino

```
    - platform: template
      sensors:
        dest_address:
          value_template: >-
             {%- if is_state("input_select.destino", "Casa")  -%}
               zone.home
             {%- elif is_state("input_select.destino", "Trabajo")  -%}
               zone.trabajo
             {%- elif is_state("input_select.destino", "Casa Jose Luis")
-%}
               zone.casa_jose_luis
             {%-  elif  is_state("input_select.destino",  "  Pabellon
entrenamiento")  -%}
               zone. pabellon_entrenamiento
             {%- elif is_state("input_select.destino", "Local ensayo")
-%}
               zone.local_ensayo
             {%- else -%}
               Desconocido
             {%- endif %}

        orig_address:
          value_template: >-
             {%- if is_state("input_select.origen", "Casa")  -%}
               zone.home
             {%- elif is_state("input_select.origen", "Trabajo")  -%}
               zone.trabajo
             {%- elif is_state("input_select.origen", "Casa Jose Luis")
-%}
               zone.casa_jose_luis
             {%-   elif   is_state("input_select.origen",   "Pabellon
entrenamiento")  -%}
               zone.pabellon_entrenamiento
             {%- elif is_state("input_select.destino", "Local ensayo")
-%}
               zone.local_ensayo
             {%- else -%}
               Desconocido
             {%- endif %}
```

Y por último es necesario crear el sensor waze para que nos recoja los datos de las listas anteriores:

```
    - platform: waze_travel_time
      name: "Tiempo Viaje"
      origin: sensor.orig_address
      destination: sensor.dest_address
      region: 'EU'
```

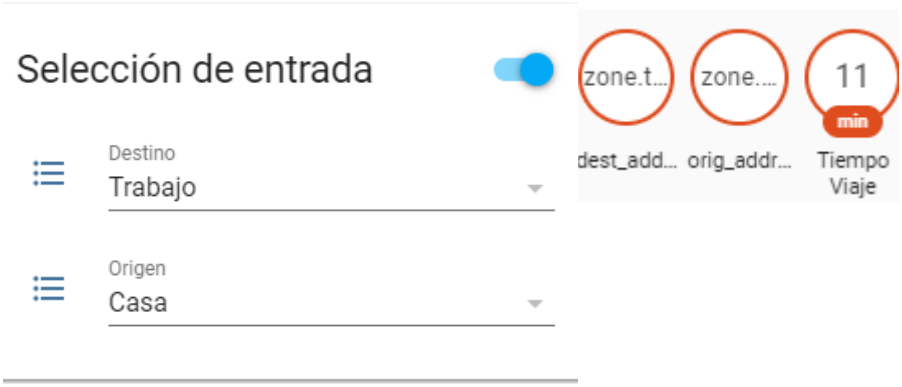


Figura 6-22. Tiempo Viaje.

6.14. Gateway Xiaomi

En este apartado voy a presentar el entorno domótico que ofrece Xiaomi, el fabricante chino que hoy en día todos conocemos por sus diferentes productos con un coste muy abaratado pero que además nos ofrece una calidad considerable. Pues bien, Xiaomi cuenta con un apartado dentro de sus productos que es la domótica, en el que podemos encontrar sensores de movimiento, de puerta y ventana, elevadores para cortinas, cerraduras, cámaras, etc…

La parte fundamental y el eje de todo es el Gateway ya que sus sensores trabajan con el protocolo de comunicación ZigBee, por lo que es necesario tener este Gateway que recoja los datos de los sensores y sea capaz de conectarse a la red de casa para la distribución de la información. En resumidas cuentas, es una pasarela zigbee.

Figura 6-23. Ecosistema xiaomi.

Antes de integrarlo en Home Assitant es necesario descargar la aplicación *MiHome* desde el *Play Store.* Una vez descargada, accedemos a ella, creamos una cuenta y seguimos los pasos paera la instalación del Gateway. Una vez instalado es necesario nos dirigimos a la pestaña *Opciones* > *About* > y pulsamos repetidas veces sobre la versión para poder autorizar el uso en la red local. Seguidamente nos aparecerá la clave para introducir en HA.

Dentro de nuestro sistema deberemos añadir las siguientes líneas en el fichero de configuración

```
xiaomi_aqara:
```

```
discovery_retry: 5
gateways:
  - key: !secrets passxiaomi
```

Una vez integrado, nos aparecerá también como una entidad de luz, ya que tiene incorporado unos leds de iluminación. También tiene una lista de sonidos a emitir mediante un altaviz que tiene integrado. En el archivo *customize.yaml* se procederá a cambiarle el nombre.

```
light.gateway_light_7811dcfb1f1b:
 friendly_name: Gateway_xiaomi
```

6.15. Sensor de movimiento

Para esta tarea es necesario tres componentes:

- Sensor de movimiento AM312
- Controlador ESP8266 (Wemos D1 mini)
- Cable dupond hembra-hembra

El ESP8266 es un chip Wi-Fi de bajo coste con pila TCP/IP completa y capacidad de MCU (Micro Controller Unit). Permite la conexión WiFi y es compatible con el protocolo TCP/IP por lo que el objetivo principal es dar acceso a una red.

La placa Wemos D1 Mini es la versión de menor tamaño de su hermana mayor Wemos D1, que veremos en una próxima entrada. Con unas dimensiones de 34.2mm x 25.6mm y un peso de 3g, es una de las placas más pequeñas basadas en el ESP8266.

De forma muy resumida estas son algunas de las principales características:

- Velocidad: 80MHz/160MHz
- Flash: 4M bytes
- Tensión funcionamiento: 3.3V
- Entradas y salidas digitales: 11, todos (salvo el D0) con PWM, interrupciones, e I2C
- Entradas analógicas: 1 (Max. 3.2V)
- Conector Micro-USB

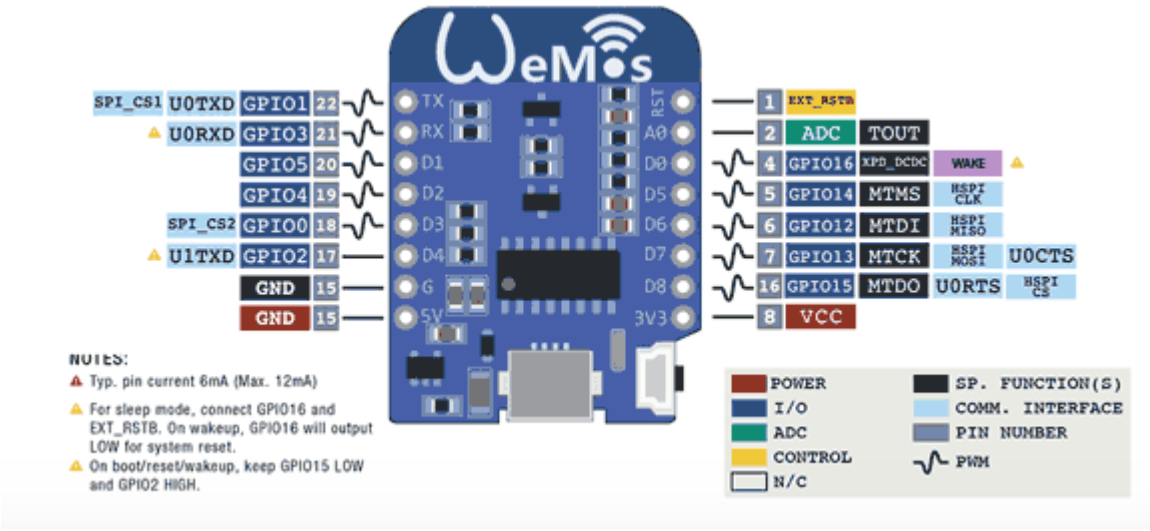


Figura 6-24. Wemos D1 mini.

Para la utilización del micro en nuestro sistema será necesario flashearlo y hacerle correr el firmware ESP Easy, que es un firmware MCU gratuito y de código abierto para Internet de las cosas. y desarrollado originalmente por la comunidad LetsControlIt.com. Para ello conectaremos el microcontrolador al pc mediante el cable USB y en la configuración de Windows veremos que puerto serie le corresponde para poder comunicarnos con él (COM 4). Se desccarga el firmware de ESP easy desde el siguiente enlace https://github.com/letscontrolit/ESPEasy/releases. El siguiente paso es hacer doble clic en el archivo flash.cmd. Se abrirá una ventana de terminal donde se nos pedirá que ingresemos el número de puerto serie (virtual) en el que está conectado el Wemos (4), La siguiente pregunta es sobre el tamaño del flash de nuestro dispositivo. En el caso del Wemos D1 mini es 4096 y después de ingresar el tamaño del flash, se nos pedirá que indiquemos qué versión de firmware cargar que será 120.

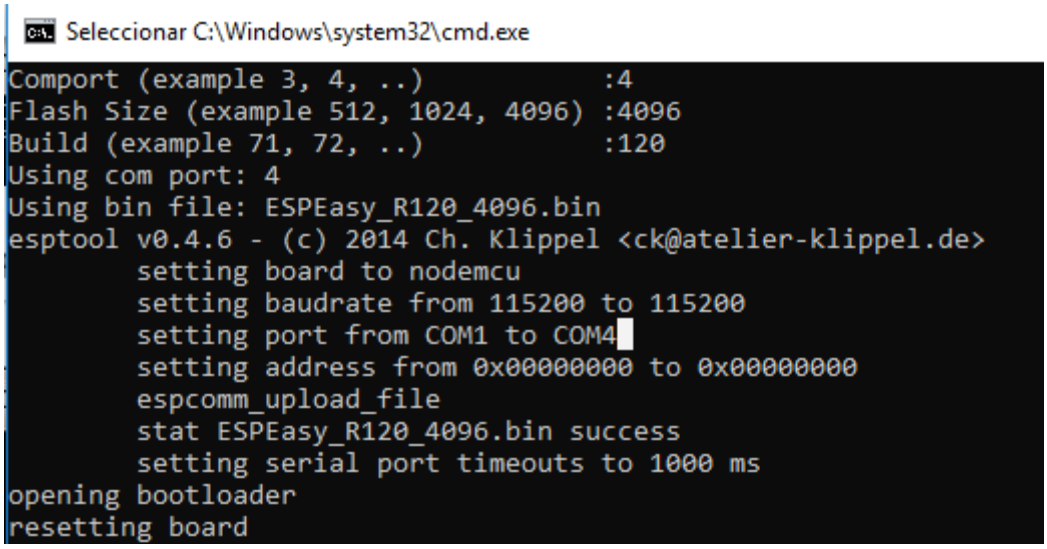


Figura 6-25. Flasheo Wemos.

Una vez terminada la carga del firmware, el micro crea una red wifi llamada *ESP_easy_0* a la que deberemos acceder con la contraseña *configesp* y se nos abrirá en el navegador una página para la configuración del dispositivo, en el que ya lo vincularemos con nuestra red wifi de casa y se le asignará una dirección IP con la

que podremos acceder para la configuración.

Figura 6-26. Acceso Wemos.

Antes de proceder a la propia instalación del sensor de movimiento se hace necesario crear un sub apartado que nos hable de MQTT y su agregación a nuestro sistema domótico.

6.15.1 MQTT

El MQTT se trata de un protocolo de comunicaciones estandarizado, muy ligero y sencillo que nos permite comunicar de una forma rápida y sin gran consumo dispositivos de forma remota. Funciona sobre el TCP/IP, es decir, sobre las redes que habitualmente tenemos en casa y que reina en internet, por lo que podemos hacer el paso de dicha información, no solo a nivel de una red local sino dar el salto a través de internet.

Se trata de un protocolo en el que se envían los mensajes y, por otro lado, otros dispositivos están "escuchando" por si algunos de los mensajes enviados son de interés. Un ejemplo para entenderlo mejor es el que usaré en mi sistema. Tenemos un dispositivo (sensor de movimiento) que está en una zona en la cual cuando detecta movimiento informa con MQTT con un mensaje sencillo del estilo a "/sensormov/sensormovimiento/Switch 1", otro dispositivo, podría estar pendiente de los mensajes que vayan etiquetados con ""/sensormov/sensormovimiento/Switch" con el fin de recoger el dato que le manda, en nuestro ejemplo, 1, para ver si realiza determinada acción.

Para poder hacer el paso de mensajes, es necesario instalar un software llamado broker que es el encargado de recibir los mensajes, así como aceptar suscripciones de otros dispositivos (los que están "pendientes" de ciertos mensajes) para mostrarles los mensajes relacionados con su interés. En nuestro caso ese bróker será *mosquitto* y operará por efecto en el puerto 1833.

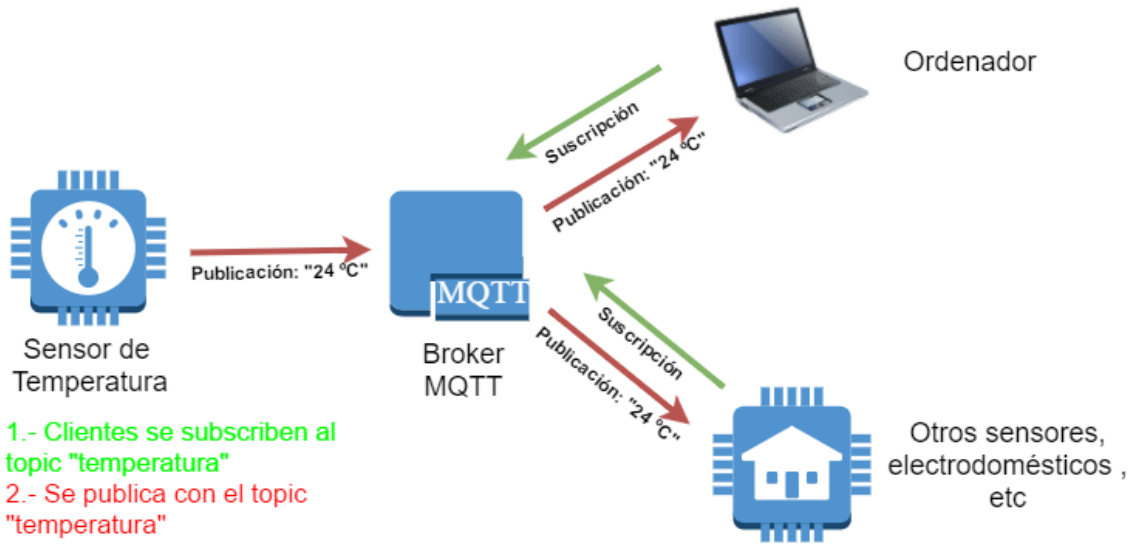


Figura 6-27. MQTT.

Gracias a este protocolo, los microcontroladores como el ESP8266 (lo usamos en este apartado), son capaces de funcionar de manera bidireccional (ya que mandan y se suscriben en el broker) mensajes para el control de diferentes dispositivos.

Mosquitto se encuentra instalado en Hassbian pero no activado, por lo que habrá que escribir el siguiente código: `sudo hassbian-config install mosquitto`. En su instalación nos pedirán un nombre y contraseña que utilizaremos más adelante.

```
Operation completed...

Your MQTT broker is running at 192.168.0.200:1883 or if preferred hassbian.local

To continue have a look at https://home-assistant.io/docs/mqtt/
```

Figura 6-28. Broker mosquitto.

Para incorporarlo e HA es necesario añadir las siguientes líneas al fichero de configuración.

```
mqtt:
  broker: 192.168.0.200
  port: 1883
  keepalive: 60
  username: santihome
  password: passbroker
```

Una vez se conoce mqtt y tenemos el broker instalado podemos proceder a la configuración e instalación del sensor de movimiento. El conexionado es muy sencillo, en el sensor tenemos 3 pines que iran conectados a la alimentación de 5v del ESP8266, a GND, y al pin D5.

Para configurar el sensor de movimiento en ESPeasy accedemos a la dirección 192.168.0.167 y tenemos una interfaz donde poder hacer los cambios. Una vez que accedemos a la IP, entramos en la pestaña Devices, pulsamos en la primera linea que esté libre, ya que podemos tener muchos dispositivos más conectados, y le

damos a Edit.

Una vez dentro, en el desplegable, elegimos "Switch input – switch", ahora nos quedaría configurar el nombre, podemos poner, por ejemplo "sensormovimiento", marcamos Enabled y tendremos que decirle en que patilla hemos conectado la central del sensor de movimiento, en nuestro ejemplo, D5. Marcamos también, "Send boot state" si queremos que mande la señal de cómo está cuando arrancamos y "Send to Controller" para que lo mande al servidor MQTT que tenemos que tener configurado. En la pestaña controllers indicaremos la dirección del servidor mqtt (192.168.0.200) y la contraseña que utilizamos en su creación.

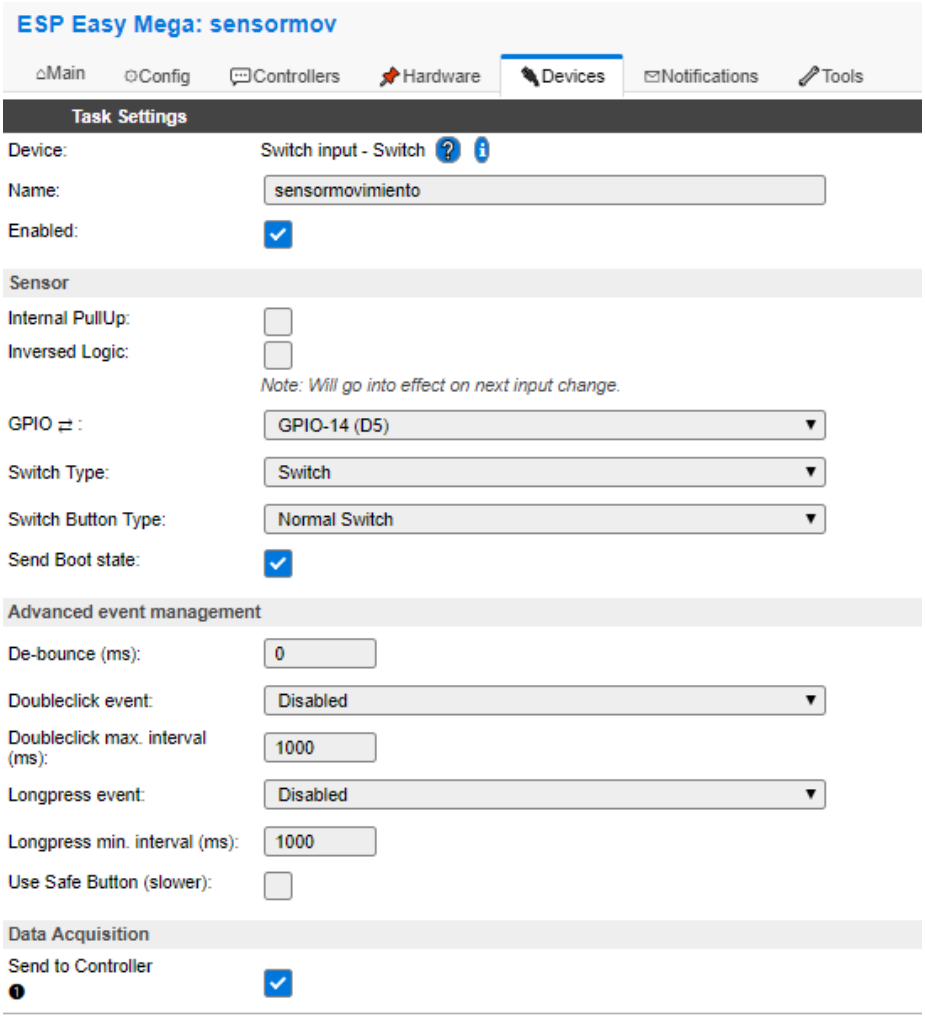


Figura 6-29. ESP Easy.

Por último, añadimos el siguiente código en nuestro fichero de configuración

```
binary_sensor:
  - platform: mqtt
    name: "Sensor movimiento"
    state_topic: "/sensormov/sensormovimiento/Switch"
    payload_on: "1"
    payload_off: "0"
```

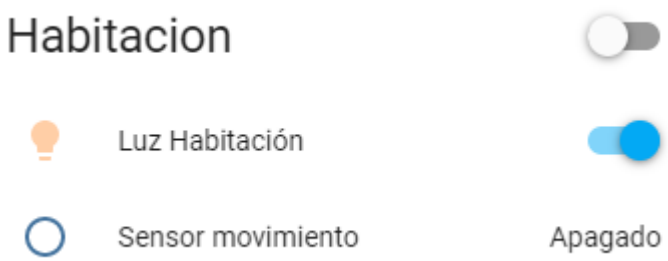


Figura 6-30. ESP Easy.

6.16. Escenas

El concepto de escena que es más que la captura en los estados que desea que estén ciertas entidades. Por ejemplo, una escena puede especificar que la luz A debe estar encendida y la luz B debe ser de color rojo brillante.

Un ejemplo de escena que voy a usar en mi sistema es el tipo de iluminación que se quiere en el salón según la persona que a la hora de ver una película, en la que se incluye el color mediante la propiedad *color_name* y el brillo en porcentaje mediante la propiedad *brightness_pct.*

```
scene:
  - name: peli_marisa
    entities:
      light.salon:
        state: true
        color_name: hotpink
        brightness_pct: 50
  - name: peli_santi
    entities:
      light.salon:
        state: true
        color_name: darkorange
        brightness_pct: 25
  - name: peli_emi
    entities:
      light.salon:
        state: true
        color_name: lightskyblue
        brightness_pct: 75
```

Figura 6-31. Escenas.

6.17. Automatizaciones

En este apartado se crearán distintas automatizaciones para dar funcionalidad a los distintos dispositivos que tenemos instalados, y poder dotar el sistema de algo más de inteligencia aprovechando así esta maravillosa capacidad que nos ofrece Home Assistant. Ya se crearon algunas automatizaciones a lo largo de este apartado como por ejemplo la de notificar con la temperatura de la CPU o el control de precios en GearBest. Para una estructura más ordenada el fichero el código correspondiente a las automatizaciones se incluirá en el fichero *automations.yaml*

6.17.1 Notificacion Versión HA

Esta automatización nos avisa a telegram que debemos actulizar HA.

```
- id: 'Notificacion Nueva Version'
  alias: Notificacion Nueva Version
  trigger:
  - entity_id: sensor.ha_version_update
    from: 'no'
    platform: state
    to: si
  condition: []
  action:
  - data:
      message: HA necesita actualizarse
    service: notify.telegramsanti
```

6.17.2 Luces Matinal

Encendido de luces al despertar. En el disparador utilizo la hora a la que me levanto por las mañanas, y en las condiciones se comprueba que sea entre semana y que haya salido el sol, ya que según la época del año la hora en la que amanece es diferente y hacen falta, o no, las luces.

```
- id: 'Luces matinal'
  alias: Luces Matinal
  trigger:
  - at: 06:50
    platform: time
  condition:
    - after: sunrise
      condition: sun
```

```
    - weekday:
          - mon
          - tue
          - wed
          - thu
          - fri
      condition: time
  action:
  - entity_id:
    - light.yeelight_color1_04cf8ca2deda
    - switch.sonoff_10008146b5
    - light.salon
    service: light.turn_on
```

6.17.3 Luces Off

Esta automatización se encargará que cuando todos los dispositivos salen de casa, si alguna luz está encendida la apaga. Se hace el seguimiento de los dispositivos mediante el *device_tracker* y se usa el grupo de todas las luces.

```
- id: 'Luces OFF'
  alias: Luces OFF
  trigger:
  - entity_id: group.all_devices
    from: home
    platform: state
    to: not_home
  condition: []
  action:
  - alias: ''
    entity_id: group.all_lights
    service: homeassistant.turn_off
  - alias: ''
    entity_id: switch.sonoff_10008146b5
    service: switch.turn_off
```

6.17.4 Nadie en casa

Cuando los dispositivos salen de casa y todos están en el estado "*not_home*" se manda un mensaje de notificación a telegram "no hay nadie en casa"

```
- id: 'Nadie en casa'
  alias: Nadie en casa
  trigger:
  - entity_id: group.all_devices
    from: home
    platform: state
    to: not_home
  condition: []
  action:
  - data:
      message: No hay nadie en casa
```

6.17.5 Sensor movimiento y luces

Esta automatización es creada mediante el sensor de movimiento y la luz de la cocina (que alumbra a la del pasillo y baño) para que cuando en la noche al levantar de la cama, el sensor de movimiento colocado en la habitación lo captará y se encenderá la luz y viceversa cuando se vuelva del baño. Este apartado se realiza mediante dos automatizaciones.

```
- id: '1567445371945'
  alias: Luz Noche ON
  trigger:
  - entity_id: binary_sensor.sensor_movimiento
    from: 'off'
    platform: state
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.sonoff_10008146b5
    state: 'off'
  - after: 00:00
    before: 06:50
    condition: time
  action:
  - entity_id: switch.sonoff_10008146b5
    service: switch.turn_on

- id: '1567445371946'
  alias: Luz Noche Off
  trigger:
  - entity_id: binary_sensor.sensor_movimiento
    from: 'off'
    platform: state
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.sonoff_10008146b5
    state: 'on'
  - after: 00:00
    before: 06:50
    condition: time
  action:
  - entity_id: switch.sonoff_10008146b5
    service: switch.turn_off
```

6.17.6 Vigilar habitación

Esta automatización se active cuando el dispositivo de Santi sale de casa, y el sensor de movimiento capta información, por lo que se envía un mensaje al móvil de Santi alertando.

```
- id: '1567446767124'
  alias: Santi fuera
  trigger:
  - entity_id: binary_sensor.sensor_movimiento
    from: 'off'
    platform: state
    to: 'on'
  condition:
  - condition: state
```

```
    entity_id: device_tracker.80_35_c1_60_40_58
    state: not_home
  action:
  - data:
      message: Alguien en la habitación
    service: notify.telegramsanti
```

6.18. Organización de la interfaz

Por último, y no menos necesario, es necesario tener un orden y organización que se tiene en la interfaz gráfica. Mi sistema, aunque no es muy grande, ya requiere de cierta organización para facilitar la vida al usuario, de esto se trata la domótica, de facilitar. La organización a seguir será en grupos y pestañas.

Grupo, como su nombre indica consiste en agrupar en una caja las entidades que nos interese, por ejemplo, luces de una habitación. Las pestañas son, estableciendo un símil, como las pestañas del navegador y con ellas creamos otra página dentro de nuestro HA en la que posteriormente podemos incluir los grupos que hemos creado anteriormente. La diferencia para crear un grupo o una pestaña es el true/false que hay en la propiedad *view*. Una entidad puede estar en varios grupos sin problema, es decir, se puede incluir una luz en un grupo de luces y en grupo de habitaciones simultáneamente

En mi sistema voy a crear cuatro pestañas; la primera "Mi casa" nos detalla información sobre luces, enchufes, sensores organizados por grupos con sus respectivas habitaciones. La segunda "parámetros del sistema" en las que se incluirán todo lo referente a la información del sistema de la Rapsberry y HA. En la tercera pestaña se incluirán las automatizaciones y escenas. La cuarta es llamadas "útiles" para organizar la información sin clasificar como el tiempo de viaje.

Los grupos y pestañas se crearán dentro del archivo *groups.yaml,* que este a su vez estará contenido en un carpeta *groups* y enlazado en el archivo de configuración mediante *group: !include_dir_merge_named groups*

Los grupos serán los siguientes:

- Dentro de la pestaña "Mi casa" habrá cinco grupos que serán salón, habitación, pasillo, personas y tiempo.
- Dentro de la pestaña "parámetros del sistema" se colocarán los grupos "Rapsberry" y "HA"
- Dentro de la pestaña automatizaciones y escenas, habrá dos grupos que recojan ambas.

Aquí se expone el ejemplo de la pestaña "Mi casa" y sus respectivos grupos, los descritos en los puntos de arriba se encuentran en el Anexo B.

```
default_view:
  name: Mi casa
  view: true
  icon: mdi:home
  entities:
    - group.salon
    - group.habitacion
    - group.personas
    - group.tiempo

#Aquí están los cuatros grupos de la pestá "Mi casa"
salon:
  name: Salón
  view: false
  entities:
    - light.salon
    - switch.sonoff_10008146b5
    - switch.07200091bcddc20d319a
    - sensor.dht_sensor_temperature
```

```
    - sensor.dht_sensor_humidity

habitacion:
  name: Habitacion
  view: false
  entities:
    - light.yeelight_color1_04cf8ca2deda

Pasillo:
  name: Pasillo
  view: false
  entities:
    - light.gateway_light_7811dcfb1f1b

personas:
  name: Personas
  view: false
  entities:
    - device_tracker.80_35_c1_60_40_58
    - device_tracker.c8_d7_b0_52_95_20
    - device_tracker.f4_e3_fb_79_5d_1f

tiempo:
  name: Otros
  view: false
  entities:
   - sun.sun
   - weather.santi_home
```

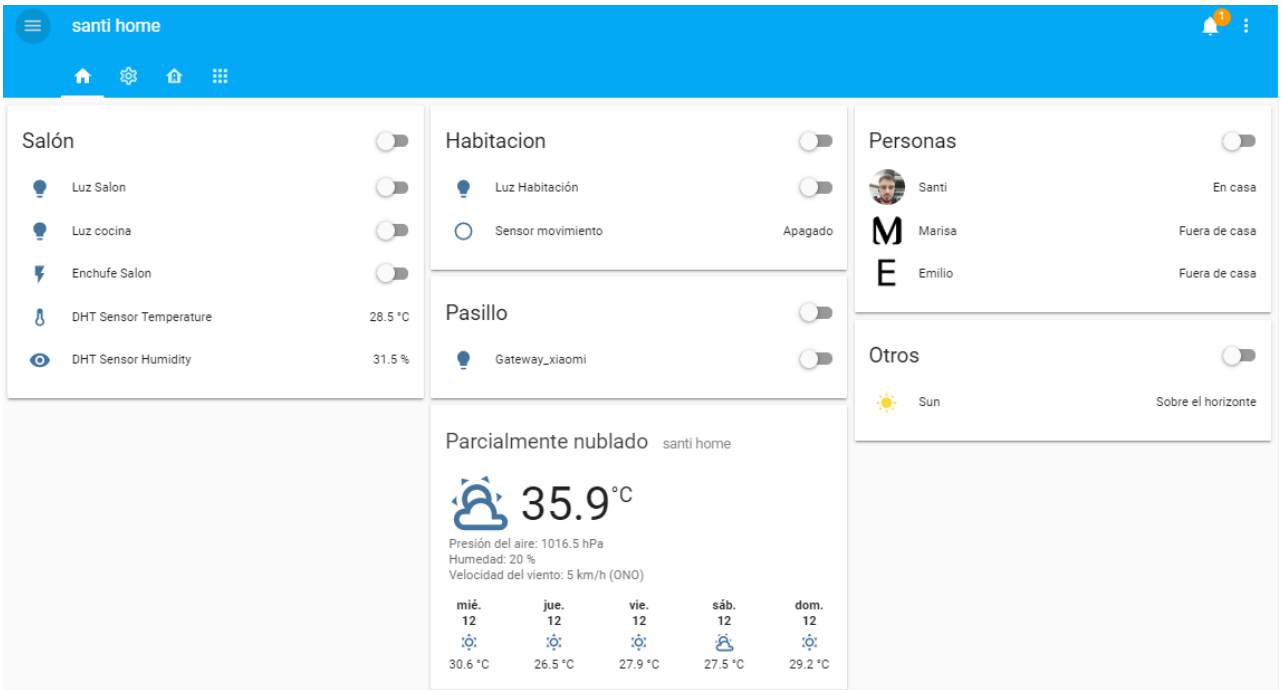


Figura 6-32. Interfaz organizada.

PRESUPUESTO

Producto	Cantidad	Precio
Raspberry Pi 3 Model B+	1	35 €
Micro SD Samsung 32GB	1	7.90 €
Sensor DHT22	1	2.21 €
Enchufe Smart Life	1	15 €
Bombilla Yeelight	1	22 €
Bombilla Smart Life	1	9 €
Sonoff Touch	1	11 €
Gateway Xiomi	1	32 €
Sensor movimiento HC-SR501	1	1.56 €
Cables	1	3.63 €
ESP8266	1	3.57 €
Total		142.57€

CONCLUSIONES

Las conclusiones que he ido sacando de este proyecto se centran básicamente en los siguientes puntos principales: el primero es el avance en materia de automatización del hogar que hemos tenido con el desarrollo de la tecnología, ya que hasta hace relativamente pocos años era impensable que pudiese llegar a casi todos los bolsillos mediante componentes baratos y plataformas de organización a mano del usuario. La segunda conclusión es que HA es un software maravilloso debido a que deja atrás la maraña de fabricantes y aplicaciones propias para la automatización del lugar, haciendo posible la integración de diferentes marcas en un mismo ecosistema. Para terminar, me encantaría decir la cantidad de información que tenemos en la web para poder aprender desde cero cualquier cosa, un ejemplo claro es este proyecto, ya que cuando lo empecé ni sabía lo que era Home Assistant y gracias a los miles de tutoriales, documentación, y foros pude construir mi propio sistema.

REFERENCIAS

- **Introducción**

[|1] Apuntes de la asignatura, *Domótica 4ª GITT*.

[2] Web de casadomo, https://www.casadomo.com/

[3] Web Universidad Tecnológica Metropolitana, http://domotica-utem.blogspot.com/

- **Plataformas domóticas**

[4] Web oficial Domoticz, https://www.domoticz.com/

[5]Web oficial OpenHab, https://www.openhab.org/

[6] Reviews tecnológicos Xataka, https://www.xataka.com/domotica-1/

- **Raspberry Pi**

[7] https://raspberryparatorpes.net/

[8] https://www.raspberrypi.org

- **Home Assistant**

[9] Conferencia de Paulus Schoutsen (creador de HA) en el Pycon 2016, https://www.youtube.com/watch?v=Cfasc9EgbMU&t=893s

[10] Transparecias de la presentación https://speakerdeck.com/pycon2016/paulus-schoutsen-awaken-your-home-python-and-the-internet-of-things?slide=3

[11] Web Oficial Home Assitant, https://www.home-assistant.io/

[12] https://www.intel.es/ (para las intels NUC)

[13] https://domoticaencasa.es/

[15] https://community.home-assistant.io

[16] Web Oficial lenguaje YAML https://yaml.org/

[17] https://nmap.org/man/es/

[18] Solución problema Sonoff, https://github.com/peterbuga/HASS-sonoff-ewelink/issues/116

[19] https://domology.es/grupos-pestanas-paneles-ha/

[20] Guía de instalación ESP, https://www.luisllamas.es/wemos-d1-mini-una-genial-placa-de-desarrollo-con-esp8266/

[21] Web oficial MQTT, http://mqtt.org/

ANEXO A

Zonas.yaml

```
- name: Trabajo
  latitude: 40.407389
  longitude: -3.788592
  radius: 120
  icon: mdi:worker

- name: Casa Jose Luis
  latitude: 40.432888
  longitude: -3.712306
  radius: 120
  icon: mdi:account-clock

- name: Pabellon entrenamiento
  latitude: 40.600782
  longitude: -3.708090
  radius: 120
  icon: mdi:tennis

- name: Local ensayo
  latitude: 40.659103
  longitude: -3.769896
  radius: 120
  icon: mdi:sword
```

ANEXO B

```
Groups.yaml
#-----------------#
# PESTAÑA MI CASA #
#-----------------#

default_view:
  name: Mi casa
  view: true
  icon: mdi:home
  entities:
    - group.salon
    - group.habitacion
    - group.pasillo
    - group.personas
    - group.tiempo


#{ Aqui están los cuatros grupos de la pestá "Mi casa"
salon:
  name: Salón
  view: false
  entities:
    - light.salon
    - switch.sonoff_10008146b5
    - switch.07200091bcddc20d319a
    - sensor.dht_sensor_temperature
    - sensor.dht_sensor_humidity

habitacion:
  name: Habitacion
  view: false
  entities:
    - light.yeelight_color1_04cf8ca2deda
    - binary_sensor.sensor_movimiento

Pasillo:
  name: Pasillo
  view: false
  entities:
    - light.gateway_light_7811dcfb1f1b

personas:
  name: Personas
  view: false
  entities:
    - device_tracker.80_35_c1_60_40_58
```

```
    - device_tracker.c8_d7_b0_52_95_20
    - device_tracker.f4_e3_fb_79_5d_1f

tiempo:
  name: Otros
  view: false
  entities:
   - sun.sun
   - weather.santi_home


#----------------------------------
#----------------------------------
#----------------------------------

#----------------------------#
# PESTAÑA PARAMETROS SISTEMA #
#----------------------------#

Parametros_sistema:
  name: Parametros sistema
  view: true
  icon: mdi:settings-outline
  entities:
    - group.raspberry
    - group.HA


raspberry:
  name: Raspberry
  view: false
  entities:
    - sensor.disk_use_percent
    - sensor.last_boot
    - sensor.memory_free
    - sensor.memory_use_percent
    - sensor.processor_use
    - sensor.temperatura_cpu

HA:
  name: HA
  view: false
  entities:
    - sensor.version_instalada_ha
    - sensor.ha_version_update


#----------------------------------
#----------------------------------
#----------------------------------

#----------------------------#
# Automatizaicones y escenas #
#----------------------------#

Automatizaciones_escenas:
  name: Aut_esc
```

```
  view: true
  icon: mdi:home-floor-a
  entities:
    - group.automatizaciones
    - group.escenas


automatizaciones:
  name: automatizaciones
  view: false
  entities:
    - automation.luces_matinal
    - automation.luces_off
    - automation.nadie_en_casa
    - automation.notificacion_nueva_version
    - automation.notificacion_temperatura_cpu
    - automation.luz_noche_off
    - automation.luz_noche_on
    - automation.santi_fuera

escenas:
  name: escenas
  view: false
  entities:
    - scene.peli_emi
    - scene.peli_santi
    - scene.peli_marisa

#---------------------------------
#---------------------------------
#---------------------------------
#----------------------------#
# OTROS                      #
#----------------------------#
utiles:
  name: utiles
  view: true
  icon: mdi:apps
  entities:
    - input_select.destino
    - input_select.origen
    - sensor.tiempo_viaje
```

ANEXO C

```
Configuration.yaml
homeassistant:
  name: santi home
  latitude: 40.616138
  longitude: -3.723874
  customize: !include customize.yaml



config:

frontend:

http:

history:
map:
logbook:

sun:

group: !include_dir_merge_named groups
automation: !include automations.yaml
script: !include_dir_named scripts

discovery:

#descubrimiento de los tre smartphones de casa
device_tracker:
  - platform: nmap_tracker
    hosts: 192.168.0.1/24
    interval_seconds: 10
    consider_home: 180
    scan_options: " -privileged -sP -host-timeout 10s "
    new_device_defaults:
      track_new_devices: False
      hide_if_away: True

sensor:
  #Sensor dht22 temperatura y humedad del salon
  - platform: dht
    sensor: DHT22
    pin: 4
    monitored_conditions:
      - temperature
      - humidity
```

```
      #Parametros para rapsberry Pi

    - platform: systemmonitor
      resources:
        - type: disk_use_percent
          arg: /
        - type: memory_use_percent
        - type: memory_free
        - type: processor_use
        - type: last_boot


    - platform: command_line
      name: Temperatura CPU
      command: "cat /sys/class/thermal/thermal_zone0/temp"
      unit_of_measurement: "°C"
      value_template: '{{ value | multiply(0.001) | round(1) }}'

    #informacion sobre la version de HA
    - platform: version
      name: Version instalada HA


    #información sobre la ultima versión de HA
    - platform: scrape
      resource: https://home assistant.io
      name: Ultima version disponible
      select: ".current version h1"
      value_template: '{{ value.split(":")[1] }}'


    - platform: template
      sensors:
          ha_version_update:
            friendly_name: '¿Nueva versión?'
            value_template: >-
              {%   if   states('sensor.Ultima_version_disponible')   ==
states('sensor.Version_instalada_HA') %}
                Sí
              {% else %}
                No
              {% endif %}

      #sensor template para los tiempos de viaje
    - platform: template
      sensors:
        dest_address:
          value_template: >-
             {%- if is_state("input_select.destino", "Casa")  -%}
               zone.home
             {%- elif is_state("input_select.destino", "Trabajo")  -%}
               zone.trabajo
             {%- elif is_state("input_select.destino", "Casa Jose Luis")
-%}
               zone.casa_jose_luis
```

```
            {%-   elif   is_state("input_select.destino",   "Pabellon
entrenamiento")  -%}
              zone.pabellon_entrenamiento
            {%- elif is_state("input_select.destino", "Local ensayo")
-%}
              zone.local_ensayo
            {%- else -%}
              Desconocido
            {%- endif %}

        orig_address:
          value_template: >-
            {%- if is_state("input_select.origen", "Casa")  -%}
              zone.home
            {%- elif is_state("input_select.origen", "Trabajo")  -%}
              zone.trabajo
            {%- elif is_state("input_select.origen", "Casa Jose Luis")
-%}
              zone.casa_jose_luis
            {%-   elif   is_state("input_select.origen",   "Pabellon
entrenamiento")  -%}
              zone.pabellon_entrenamiento
            {%- elif is_state("input_select.destino", "Local ensayo")
-%}
              zone.local_ensayo
            {%- else -%}
              Desconocido
            {%- endif %}


    - platform: waze_travel_time
      name: "Tiempo Viaje"
      origin: sensor.orig_address
      destination: sensor.dest_address
      region: 'EU'


  #Integración del enchufe del salón
  tuya:
    username: santiromerous@gmail.com
    password: !secret smartlifepass
    country_code: 34
    platform: smart_life



  #Integración interruptor cocina
  sonoff:
      username: santiromerous@gmail.com
      password: !secret sonoffpass
      scan_interval: 10
      grace_period: 600
      api_region: 'eu'
```

```
  #integración de telegram
  telegram_bot:
    - platform: polling
      api_key: 802308700:AAHItuFuLiWT-woElL1PbJK46XZVv6B07ow
      allowed_chat_ids:
        - 556769163

  #Integracion del componente notify que se apoya en la plataforma
Telegram
  notify:
    - name: telegramsanti
      platform: telegram
      chat_id: 556769163



  scene:
    - name: peli_marisa
      entities:
        light.salon:
          state: true
          color_name: hotpink
          brightness_pct: 50
    - name: peli_santi
      entities:
        light.salon:
          state: true
          color_name: darkorange
          brightness_pct: 25
    - name: peli_emi
      entities:
        light.salon:
          state: true
          color_name: lightskyblue
          brightness_pct: 75



  #input select utilizado para los timepos del viaje
  input_select:
    destino:
      name: Destino
      options:
        - Casa
        - Trabajo
        - Casa Jose Luis
        - Pabellon entrenamiento
        - Local ensayo
      initial: Trabajo
      icon: mdi:car
    origen:
      name: Origen
      options:
        - Casa
        - Trabajo
```

```
      - Casa Jose Luis
      - Pabellon entrenamiento
      - Local ensayo
    initial: Casa
    icon: mdi:car

mqtt:
  broker: 192.168.0.200
  port: 1883
  keepalive: 60
  username: santihome
  password: !secret passmqtt


binary_sensor:
  - platform: mqtt
    name: "Sensor movimiento"
    state_topic: "/sensormov/sensormovimiento/Switch"
    payload_on: "1"
    payload_off: "0"


xiaomi_aqara:
  discovery_retry: 5
  gateways:
    - key: !secret passxiaomi
```

ANEXO D

Automations.yaml

```
  - id: Notificacion temperatura cpu
    alias: Notificacion temperatura CPU
    trigger:
    - above: '65'
      entity_id: sensor.temperatura_cpu
      platform: numeric_state
    condition: []
    action:
    - data:
        message: La temperatura de la CPU ha llegado a los   {{
states.sensor.sensor.temperatura_cpu.state
          }} °C
      service: notify.telegramsanti
  - id: Notificacion Nueva Version
    alias: Notificacion Nueva Version
    trigger:
    - entity_id: sensor.ha_version_update
      from: 'no'
      platform: state
      to: si
    condition: []
    action:
    - data:
        message: HA necesita actualizarse
      service: notify.telegramsanti
  - id: Luces matinal
    alias: Luces Matinal
    trigger:
    - at: 06:50
      platform: time
    condition:
    - after: sunrise
      condition: sun
    - weekday:
      - mon
      - tue
      - wed
      - thu
      - fri
      condition: time
    action:
    - entity_id:
      - light.yeelight_color1_04cf8ca2deda
      - switch.sonoff_10008146b5
```

```
    - light.salon
    service: light.turn_on
- id: Luces OFF
  alias: Luces OFF
  trigger:
  - entity_id: group.all_devices
    from: home
    platform: state
    to: not_home
  condition: []
  action:
  - alias: ''
    entity_id: group.all_lights
    service: homeassistant.turn_off
  - alias: ''
    entity_id: switch.sonoff_10008146b5
    service: switch.turn_off
- id: Nadie en casa
  alias: Nadie en casa
  trigger:
  - entity_id: group.all_devices
    from: home
    platform: state
    to: not_home
  condition: []
  action:
  - data:
      message: No hay nadie en casa
    service: notify.telegramsanti
- id: '1567445371945'
  alias: Luz Noche ON
  trigger:
  - entity_id: binary_sensor.sensor_movimiento
    from: 'off'
    platform: state
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.sonoff_10008146b5
    state: 'off'
  - after: 00:00
    before: 06:50
    condition: time
  action:
  - entity_id: switch.sonoff_10008146b5
    service: switch.turn_on
- id: '1567445371946'
  alias: Luz Noche Off
  trigger:
  - entity_id: binary_sensor.sensor_movimiento
    from: 'off'
    platform: state
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.sonoff_10008146b5
    state: 'on'
```

```
    - after: 00:00
      before: 06:50
      condition: time
    action:
    - entity_id: switch.sonoff_10008146b5
      service: switch.turn_off
  - id: '1567446767124'
    alias: Santi fuera
    trigger:
    - entity_id: binary_sensor.sensor_movimiento
      from: 'off'
      platform: state
      to: 'on'
    condition:
    - condition: state
      entity_id: device_tracker.80_35_c1_60_40_58
      state: not_home
    action:
    - data:
        message: Alguien en la habitación
      service: notify.telegramsanti
```

ANEXO E

Customize.yaml

```
light.yeelight_color1_04cf8ca2deda:
  friendly_name: Luz Habitación
switch.07200091bcddc20d319a:
  friendly_name: Enchufe Salon
switch.sonoff_10008146b5:
  friendly_name: Luz cocina
  icon: mdi:lightbulb
light.salon:
  friendly_name: Luz Salon
light.gateway_light_7811dcfb1f1b:
  friendly_name: Gateway_xiaomi
```